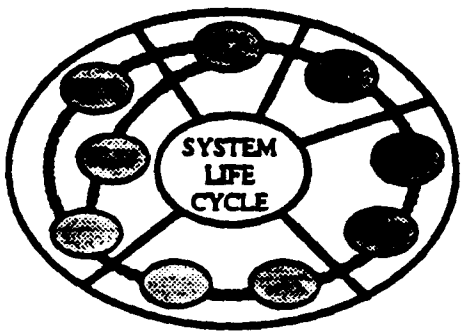
OFFICE OF SOLID WASTE AND EMERGENCY RESPONSE (OSWER)



SYSTEM LIFE CYCLE MANAGEMENT GUIDANCE

Part 3: Practice Paper

Expert Systems

January, 1989

TABLE OF CONTENTS

Chapter 5 Definition and Design Phase

Chapter 6 Development Stage

Chapter 7 Implementation Stage

Chapter 8 Operation Phase

CHAPTER 1
EXPERT SYSTEMS DEFINED

1.0 INTRODUCTION

The purpose of this practice paper is to help software developers and project managers in EPA's Office of Solid Waste & Emergency Response (OSWER) understand the issues involved in successfully applying expert systems technology to OSWER information management problems.

This paper explains when and where expert systems can be used and how the development of expert systems should be managed in OSWER. It is an extension of the *OSWER System Life Cycle Management*. The paper focuses on the expert system life cycle as it relates to the conventional life cycle phases. For this *Practice Paper* to be used effectively, the project leader and developer should have a firm understanding of the *OSWER System Life Cycle Management Guidance*, or *LCM Guidance* as it will be called elsewhere in this document. Further, this paper does not explicitly define or describe in detail how to build an expert system. The reader is responsible for obtaining this information through formal training, seminars, computer literature (see the bibliography in Section 1.8), or other media. The end of CHAPTER 1 has a *Quick Briefing on Expert Systems*.

The paper is divided into the eight chapters. Those specifically related to the *LCM Guidance* (CHAPTERS 3 through 8) should be read in sequence. When reading the chapters, bold-face terms are products associated with each of the LCM phases. The first occurrence of terms defined in the glossary are bracketed like this [term]. Each chapter has several additional features in common with the rest:

- An introduction;
- A statement of objectives;
- A list of the decisions that must be made in the life cycle phase;
- A list of success factors; and
- A description of activities that are to be conducted during each phase or stage.

Each chapter contains an illustration of the objectives, decisions, and products for the phase. The illustration at the start of each chapter provide the reader with an orientation point in each chapter.

The chapter you are reading, *Expert Systems Defined*, introduces the concept of [expert systems]. It explains the capabilities, benefits, and limitations that may be associated with their use. It provides examples of typical expert system applications.

CHAPTER 2; Implications of Using Expert Systems, gives an overview of the possible impacts expert systems could have on OSWER. It identifies possible legal and liability issues as well as potential impacts on Agency policies. The chapter describes the cross-cutting project management issues that are addressed in multiple phases of the system life cycle. These are project management plan, project participation, reviews and quality assurance, project approvals, configuration management, data administration, methodologies and [tools], cost-benefit analyses, and [knowledge management]. Finally, the chapter addresses the resources required throughout the expert system life cycle.

CHAPTER 3; Initiation Phase, describes the tasks involved in problem definition and in determining the need for an automated solution. This chapter explores characteristics of a problem that suggest a possible expert system solution.

CHAPTER 4; Concept Phase, depicts the identification of a feasible, timely, cost-effective solution to the problem. This chapter contains information on the use of proof-of-concept prototyping to refine the solution, [knowledge representation], and management techniques, expert system [control structures], and system development justification approaches. Also, it specifies required qualifications for system developers.

CHAPTER 5; Definition and Design Phase, discusses issues as they relate to confirming the suitability of the [System Concept] and determining detailed functional requirements. It discusses all aspects of selecting a development environment including [knowledge base] creation, migration to delivery environments, and user interfaces.

CHAPTER 6; Development Stage, describes issues in developing the expert system. This chapter introduces various [knowledge acquisition] methodologies, sources, and [conflict resolution] techniques. Also, it defines the means and importance of testing and validating the system.

CHAPTER 7, Implementation Stage, identifies the strategies for distributing expert systems. It includes the issues of beta testing, user registration, hardware and software requirements, training, licensing, documentation, configuration management, and version control.

CHAPTER 8, Operation Phase, focuses on the Production, Evaluation, and Archive Stages of the life cycle. It describes expert system maintenance, end-user support requirements, knowledge revalidation options and maintenance, ongoing training and documentation, and software updates.

The paper concludes with two appendices. Appendix A describes the process of evaluating [rule-based] expert system software packages (e.g., [shells]) and provides an evaluation form. This appendix will be useful to project managers and software developers during the [Definition and Design Phase] when the development environment may be a rule-based shell. The process of evaluating shells using other knowledge representation schemes (e.g., [frames], object-oriented programming, etc.) has not been defined. This type of information is widely available in the professional literature. See Section 1.7 for references on further reading. Appendix B is a glossary that provides a definition of the [artificial intelligence] terms and concepts used throughout this practice paper.

1.1 EXPERT SYSTEM CAPABILITIES

In order to understand how expert systems can be used advantageously, the project manager or software developer must first understand how knowledge processing in expert systems differs from conventional data processing. Expert systems are unique in their ability to process [knowledge], not just data. Knowledge processing differs from data processing in the type of information used, the techniques used to analyze the information, and in the form that the results of the knowledge processing are presented to the [user].

Conventional systems limit the developer to data representation using only numbers and text. They process data using complex [algorithms] that complete a discrete number of steps to reach a predetermined conclusion. Expert systems permit knowledge representation -- the encoding of human decision-making processes using symbolic terms or [symbols]. Because expert systems process knowledge, they are often referred to as [knowledge-based systems].

The ability to represent knowledge in symbolic terms expands the range of analysis techniques that computers can apply to information thus enabling a system to emulate some aspects of human performance. The expert system uses problem solving procedures such as pattern-matching to reason about the symbolic terms. An example of [symbolic reasoning] by an expert system that determines the daily forecast is, "if the sky is cloudy, then the forecast might include rain." This phrase could be used within an expert system. A conventional system would require that the symbolic terms such as "sky" and "cloudy" be defined concretely as numbers or text. The expert system uses pattern-matching on the phrases "sky is cloudy" and "forecast might include rain" to reach a conclusion about the forecast without requiring definition of each of the terms within the phrases. A conventional system would require that the developer define a set number of steps to determine the daily forecast. The expert system examines all possible solutions using the problem solving procedure that has been programmed into it.

The combination of problem solving procedures that are built into expert systems, together with the developer's ability to define problems using symbolic terms, gives expert systems the capability to store and manipulate more complex relationships between individual pieces and groups of information than can be accomplished with the processing supported by conventional systems.

In addition to knowledge representation, expert systems also provide the capability to simplify the user/computer interaction. Expert systems can be programmed to employ more of the conventions that people use when communicating with each other. Expert systems can be designed with the ability to explain the "reasoning" used in reaching a recommendation and to justify their approach to a problem, much as people do. The more sophisticated expert systems employ a [natural language] processor to permit users to consult with the system using near-English rather than structured query languages or menus.

1.2 EXPERT SYSTEM BENEFITS

Expert system applications take advantage of the above capabilities in two ways. First, information becomes more accessible so people can make better decisions. Second, where useful information is accessible, people can be more productive OSWER can be more productive in the use of its personnel, funding, and information resources through the application of expert systems. The two major benefits associated with expert systems are better decision making and increased productivity. These benefits are described below.

1.2.1 Better Decision Making

Expert systems improve the quality of decision making by providing a mechanism for pooling the knowledge of multiple experts and making that knowledge available to a wider audience. This leveraging of knowledge results in improved quality of complex work products such as permits, technical reports, and analyses that recommend actions.

Expert systems establish a basis for defensible decision-making by capturing and applying knowledge in verifiable form. For example, in developing work products such as management reports and environmental analyses, a given set of inputs, no matter how complex, should result in consistent results given closely similar data, advice, or recommendations. In addition, the process of reaching a conclusion can be explicitly demonstrated. This will ensure consistency in many decision-making activities.

1.2.2 Increased Productivity

Expert systems offer significant, measurable increases in productivity by effectively capturing the knowledge of experts and by minimizing the loss of [expertise] and knowledge due to attrition. Expert

systems provide a means of extracting, storing, and sharing knowledge in a variety of disciplines. Thus, more people have access to expertise. In turn, the experts are freed from relatively mundane tasks to focus on demanding ones.

1.3 EXPERT SYSTEM LIMITATIONS

Expert systems provide valuable new capabilities, but they also have clear limitations. As with all new technology, developers must weigh the limitations associated with the use of expert systems technology against benefits. Because expert systems emulate human performance in decision making, they may be incorrectly thought of as having the capacity to make independent judgments. Expert systems are capable of communicating advice that has been coded into them. They are not capable of producing independent decisions. Their application is limited to strictly defined domains (i.e., areas of expertise where boundaries on what expertise should be included in the system can be defined); their performance degrades dramatically when dealing with information that is beyond those boundaries.

Expert systems can manipulate only symbolic information, that is, all "real" information that is collected by observing an event. For instance, temperature and humidity in the case of weather forecasting must be translated into a form acceptable to the expert system. Any errors and biases incorporated in the translation process will be accepted by the expert system without question. An example of a translation bias is if temperature measurements input to the system are in Fahrenheit when the logic or knowledge encoded into the system is based on Celsius measurements, then the conclusions reached will be invalid.

Another limitation is the difficulty developers might experience when linking expert systems to conventional data processing systems. This would involve building a hybrid or embedded system. New expert system products are appearing regularly to bridge the gap between expert and conventional systems. However, until a greater number of these products exist, software developers will experience difficulty making expert system software communicate with conventional software.

1.4 DIFFERENCES BETWEEN DECISION SUPPORT SYSTEMS, MANAGEMENT INFORMATION SYSTEMS, AND EXPERT SYSTEMS

Based on an understanding of the capabilities, benefits, and limitations of an expert system, the project manager or software developer needs to consider the differences between expert systems and conventional systems such as database management systems. The following discussion will provide background needed to determine whether expert systems are the technology of choice for an application under consideration.

All software tools and applications bear fundamental structural similarities to each other; their differences lie in the capabilities associated with each and in the methods or mechanisms used to achieve those capabilities. Expert systems can be distinguished from conventional systems based on those differences. For example, conventional systems do not have the capability to handle uncertainty. They are developed using high level programming languages such as COBOL or Fortran, and these languages must explain in exacting detail how to solve a given problem. They provide no capacity for vagueness or uncertainty.

Expert systems accommodate uncertainty through weighting schemes. These weighting schemes calculate certainty levels by determining if a piece of information is correct or valid and increment the level of certainty accordingly. The inability to process uncertain data confines the power of conventional systems to linear programming tasks, that is, tasks where steps are performed in sequence regardless of the inputs, and which are number- or text-intensive. While conventional systems are developed using conventional languages, expert systems are often developed with software referred to as "shells." A shell provides a framework to build expert systems. Appendix A provides a framework for evaluating shells.

Expert systems are also developed using LISP or [PROLOG]. These are the recognized computer languages of artificial intelligence (AI). These languages simplify the expert system development process because they are interactive, and they provide feedback to the programmer as the program is being written. This capability allows the programmer to see results more quickly. Additionally, LISP and PROLOG code is generally more like English and can be understood more readily than can the standard code of high level languages. AI languages give the programmer the added capability of more flexibility. Whereas LISP and PROLOG permit programmers to tell the computer what to do, conventional languages require them to tell the computer how to do it. With LISP and PROLOG, less development time is spent defining each piece of information that might enter the system. The distinction between use of programming languages and expert system shells is that the [inference engine] in the shell has already been validated by the vendor. Shells are good places to start prototyping work, and the developer can move to a programming language if the complexity of the task warrants building the inference engine from the ground up.

There are additional differences between expert and conventional systems. These are described below.

1.4.1 Decision Support Systems

Decision support systems (DSS) allow users to combine their judgment with data to manipulate the data and produce reports. The users' judgement is translated into an algorithm that is then programmed into the decision support system using conventional programming languages such as COBOL or Fortran. The algorithm typically gives the user flexibility to define data manipulation

scenarios and report formats. A specialized interface allows the user to easily define meaningful data manipulation scenarios without requiring an understanding of the underlying data structures.

DSSs employ a variety of information management technologies to solve structured or unstructured problems and support managerial judgment, thus improving the effectiveness of decision-making. Typical DSS tools include spreadsheet, geographic information systems, modeling packages, and statistical packages. These systems are also linked to the functions of 4th Generation" relational database management software.

Expert systems are a specialized subset of DSSs that employ symbolic reasoning to manipulate data and produce reports. If the judgement employed by the users is procedural in nature (i.e., requires a set number of steps to reach a predetermined conclusion), then a conventional DSS should be developed. However, if the user reaches a conclusion based on a variety of factors that cannot be captured using an algorithm, then an expert system could be developed.

1.4.2 Management Information Systems

Management information systems are computer-based information systems that provide information to support management activities and functions. They support four general types of activities: transaction processing such as payroll, sales order, and inventory; operational control which ensures that activities are carried out efficiently and effectively; management control which measures performance, deciding on control actions, formulating new decision [rules], and allocating resources); and, strategic planning which involves developing strategy with which to meet organizational objectives.

Executive information systems (EIS) are a specialized form of management information system. They offer top level managers access to corporate information summarized in a manner specified by the executives so they get what they need in the required form and detail.

Management information systems do not typically have symbolic reasoning capabilities. The graphic displays are static in the sense that there is no [explanation] or interpretation facility. The user must know how to format queries; the system cannot reformat queries if the user is not sufficiently specific. Vague or uncertain information contained in a management information system query may result in computer resource intensive processing caused by extensive database [searches]).

1.5 EXPERT SYSTEM COMPONENTS AND ARCHITECTURE

Expert systems can be viewed as having three unique, isolated components. See Section 1.8 for a graphic representation of the components of an expert system. These are the user interface, knowledge base, and inference engine. Most expert systems and system development tools have these components. If project managers and software developers are aware of each component during the Definition and Design Phase of the system life cycle, a development environment can be selected that supports the components and level of complexity desired for the system.

1.5.1 User Interface

The user interface permits the user and system to communicate. The user interface may be simple. It can consist of questions posed to the user or optional responses from which the user makes a selection by typing in the response desired; or menus from which selections are made using the cursor keys or a mouse. A sophisticated user interface may employ a natural language processor written in a higher level language to ask and answer questions, to explain or justify the system's conclusions, and to accept modifications to the knowledge base.

1.5.2 Knowledge Base

The knowledge base contains the knowledge or expertise of the domain which is the designated area of expertise for the system. In many rule-based expert systems, the knowledge is encoded into sentences using an "If PREMISE, then CONCLUSION" format. [Facts] may also be stored in the knowledge base using a variety of formats such as STRONG_WIND = TORNADO. Rules are used to encode knowledge that includes reasoning about the inputs to the system, or facts already encoded into the knowledge base. Facts are used to encode passive knowledge that is concrete. For example, each entry in a table that translates temperature readings from Celsius to Fahrenheit could be included in a knowledge base as a fact.

1.5.3 Inference Engine

The inference engine contains the specific procedures and algorithms for using the rules and facts in the knowledge base to solve a problem. One possible procedure or strategy used by an inference engine in a rule-based expert system is [backward chaining]. Using backward chaining, the inference engine first considers a primary goal. The text of the goal is matched against the conclusion. For example "if the sky is cloudy, then the forecast might include rain". In this example "sky is cloudy" is a premise and "the forecast might include rain" is a conclusion. Rule matching determines whether that rule will contribute information to the resolution of the goal. If the conclusion of a rule matches the goal, then the premises of the rule are considered in turn. Each of the premises is considered to be an intermediate goal. Results of evaluating each goal are stored in memory to be

used when evaluating subsequent goals. This inference strategy is not the only inference strategy. Refer to Section 1.8 for sources of additional information.

1.6 APPLICATION TYPES

The ability of expert systems to reason about rules as well as input data makes them particularly adept at handling many different types of problems. This Section describes the types of problems and resulting applications that are well suited to the use of expert systems technology. The generic "types" of expert systems discussed below are well documented in the literature about expert systems.

1.6.1 Diagnosis and Classification

Diagnosis and classification expert systems select an answer from a fixed set of alternatives on the basis of information input to the system while it is reasoning. Much notable work has been done with expert systems in the field of diagnostics, where problems with an object (animate or inanimate) are diagnosed from observations. Medical examinations and electronic circuit board analysis have been successfully emulated by this type of system. Diagnostic systems must be able to handle intermittent symptoms, causes of faults hidden by non-related symptoms, and sometimes inconsistent models of complex systems.

Stanford University's MYCIN has performed extremely well in diagnosing bacterial infections in humans and recommending antibiotic therapy. Another diagnosis and classification expert system, MUDMAN, diagnoses problems with "mud" used in NL Baroid's oil well drilling and recommends new compositions.

1.6.2 Data Analysis and Interpretation

Data analysis and interpretation systems select a hypothesis based on measurement data and corollary information. They infer situation descriptions from accumulated sensor data. Artificial vision, image analysis, and surveillance all employ expert systems, although work is just beginning in these areas. (Note: this guidance document does not address the more esoteric research areas current in artificial intelligence.) Expert systems can deal with incomplete or contradictory data. In addition, their explanation mechanisms can demonstrate the reasoning that lies behind a complex interpretation. Expert systems are often good at assisting managers in managing complex and incompatible data sources in order to consistently reach a known range of conclusions.

An example of such a system is DIPMETER ADVISOR, developed by Schlumberger, which analyzes data from oil well instruments. Stanford University's DENDRAL analyzes molecular structures based on mass spectrogram

data. Another example of a data analysis and interpretation system, PROSPECTOR, was developed at Stanford Research Institute to interpret data about ore-grade deposit sites.

1.6.3 Design and Synthesis

Design and synthesis expert systems configure objects such as computer systems on the basis of a set of alternate possibilities. The expert system incorporates constraints that the system must meet as well as guidance for steps the system must take to meet the user's objectives. Design expert systems are used in configuring large mainframe computers, in designing circuit boards, and in preparing large annual budgets. Expert systems permit the exploration of the consequences of proposed design changes prior to implementing them. They also facilitate the subdivision of large problems into smaller ones and simplify coordination among the subsets.

Other examples include Honeywell's SYSCON configures their DPS 90 mainframe computers before assembly at the factory, and Digital Equipment Corporation's XCON which designs large VAX computers. HI Class, developed by Hughes Aircraft, solves circuit board assembly problems.

1.6.4 Prediction and Simulation

Prediction and simulation expert systems forecast what will happen on the basis of current information by depending on experience or employing models or simulation. Situations involving prediction are well suited to expert systems. Crop management and weather forecasting are two areas of current interest. Predictor systems must be able to model the ways actions change over time and to manipulate events that are ordered in time. They must also be able to deal with incomplete information, generate multiple scenarios, and use diverse data sources

For example, the USDA's COMAX advises farmers on irrigation, fertilization, and when to harvest. Purdue University's GRAIN MARKET ADVISOR helps farmers determine the best way to market the grain they produce.

1.6.5 Monitoring

Monitoring expert systems obtain data on an ongoing situation following its predicted or intended progress and alerting the user or system if there is a departure from the expected or usual. Expert systems capable of monitoring their environments compare observations to desired outcomes and report discrepancies. They must recognize alarm conditions in real time and avoid false reports of problems or emergencies. Note that "real time" expert systems are significantly different than those which allow the luxury of reflection. Because of varying environmental factors, monitoring systems must vary their anticipation of alarm conditions with time and situation.

Air traffic control and nuclear power plant management are two current fields of application. Additionally, NASA's NAVEX monitors controls on space shuttle flights.

1.6.6 Instruction

Instruction expert systems are built depending on user needs to give advice, furnish information, or perform various subtasks. Expert systems that can serve as an on-line tutorial aid to students and diagnose areas of deficiency involve the automation of instruction. These systems construct a hypothetical description of a student's knowledge in order to interpret the student's behavior. They identify remedial courses of action and develop tutorial modes of communicating the remedial action to the student. They can be used to tutor novices and to advance the development of human experts.

For example, DOPPLER DIAGNOSIS, developed by Dr. Evlin Kinney, helps train doctors in the use of non-invasive echo effect equipment. CBT ADVISOR, developed by Courseware, helps determine the suitability of instructional units for computer based training.

1.6.7 Planning

Planning expert systems select a series of actions from a complex set of alternatives to meet a user's goals. Because time and resource constraints may not permit all goals to be met, the most desirable outcome is sought. Planning systems are used in contingency environments. A planning expert system carries out prepared plans of action and can be used in a time critical situations such as responding to a natural disaster. Priorities must be established in order to resolve conflicts among goals and subgoals may need to be established to simplify complex interactions.

IBM's UNIT COMMITMENT ADVISOR assists in the shut down of power plants. Oak Ridge National Laboratories has experimented with several shells with hazardous spill containment.

1.6.8 Control

Control is a combination of monitoring a system and taking appropriate actions. Control systems interpret, predict, repair, and monitor system behavior. Thus, they incorporate many of the other problem areas presented here. Problems addressed by control systems include battle management, mission control, and business management.

Hitachi's TRAIN BRAKING ADVISOR regulates locomotive braking for accuracy and comfort. COMPONENT IMPACT ANALYSIS SYSTEM, developed by Argonne National Laboratories, advises nuclear power plant operators on valve and switch settings.

1.6.9 Repair

Debugging and repair systems specify remedial plans and apply them in limited areas. Computer programming is the most obvious area of application, but medical diagnostic systems also have debugging aspects. Expert systems could greatly enhance expertise in automotive, avionics, network, and computer maintenance.

Troubleshooting Aid for F6502, developed by Tektronix, assists technicians in the repair of F6502 instruments. Stone & Webster's PUMP PRO advises on preventive maintenance of centrifugal pumps. Honeywell's PRESS debugs operating systems software. MIT's Programmer's Apprentice assists in automatic code generation and in isolation of logical inconsistencies in programs.

1.7 PROTOTYPING OVERVIEW

Because prototyping is frequently used in expert system development, the concept is introduced here and developed further in the relevant stages.

Prototyping is inherently easier with expert systems than with building most conventional programs. Building an expert system is an incremental process; at any time during development, the partially built system will function to a limited degree. The scope and capability of the system will be limited to the depth and breadth of knowledge that has been applied to the knowledge base. When the expert system encounters a case about which it has not been given any knowledge, it simply will not offer advice or it may offer nonsense advice. With most conventional programs during the development stage of the conventional system's life cycle, if they start to function and encounter an area where logic has not been coded, they will usually end without warning or explanation. Because partially completed expert systems will run, developers can demonstrate the system as it is being developed, thereby soliciting comments, suggestions, and feedback.

Expert system development lends itself better to a detailed prototyping cycle than to a long design period for several reasons. First, experts prefer to analyze a working system rather than one on paper. Second, users are more motivated when they see a working system. Third, experts can better cite exceptions to rules that pertain to realistic problems. Exceptions to rules tend to emerge as problems are encountered. There are several categories of prototypes, each with its own role in expert system development. They are described in Sections 1.7.1 through 1.7.4.

1.7.1 Proof-of-Concept-Prototype

The Proof-of-Concept Prototype is a small working system designed to provide a preliminary feasibility assessment of the emerging solution before additional resources are allocated. It is used primarily as a proof-of-

concept test to show inherent strengths and weaknesses of the concept. It is most commonly built in the Concept Phase, and further refined in the Definition and Design Phase. This type of prototype may consist of "disposable code" or code that is used only to prove that an idea or concept is feasible.

1.7.2 Demonstration Prototype

If the System Concept meets all requirements, the Demonstration Prototype is constructed during the Definition and Design Phase. Its purposes are to codify core knowledge and to verify the knowledge representation schemes and control structures. The Demonstration Prototype is generally small and specialized, based on a narrow subset of the overall problem domain. If major revisions are needed either to the knowledge representation or control strategies, they are made at this point. Should a different approach appear to be more workable, the Demonstration Prototype can be reevaluated.

1.7.3 Full Prototype

During the Definition and Design Phase, the Full System Prototype is developed to encompass the entire problem domain. The purpose is to verify the problem domain, the knowledge representation schemes and control structures, data requirements, and interfaces. If the problem is properly scoped, the Full Prototype will be of a manageable size. The prototype contains most of the core and supporting knowledge and the agreed-upon knowledge representation and control structures. If major revisions are needed to the knowledge base, the knowledge base, or the control structures, a new approach can be designed following the guidelines in Chapter 5, Definition and Design Phase.

Prototyping is an exploratory process. If at this point the design is no longer valid, the prototype can be reevaluated based on the new information and redesigned as appropriate. The efforts that went into the development are not lost; they represent the design phase in conventional systems.

1.7.4 Production System

Finally, the Production System is built during the Development Stage. This is the actual system that will go out into the field, including all user interfaces and links to external databases. After complete testing and validation, any disclaimers about the use and liability of the expert system are added to the system.

1.8 SOURCES OF ADDITIONAL INFORMATION

1.8.1 Books

Harmon, P. and King, D. *Expert Systems: AI in Business*, John Wiley and Sons, New York, NY, 1985. This beginner's primer on expert systems does not assume any background in computer science. or AI. Contains an excellent bibliography.

Hayes-Roth, F., Waterman, D., and Lenat, D. *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983. Comprises a good description of the [architecture] of an "ideal" expert system and compares and contrasts an implementation in several early shells.

Waterman, D. *A Guide to Expert Systems*, Addison-Wesley, Reading, MA, 1986. A well-written introduction to ES technology. Includes an extensive bibliography of expert systems in various application areas.

1.8.2 Organizations, Journals, and Magazines

AI Expert, 500 Howard Street, San Francisco, CA 94105. This is a popular magazine for people interested in AI. The articles tend to be in the vein of BYTE magazine, although some deal with industry trends and applications.

American Association of Artificial Intelligence (AAAI), 445 Burgess Drive, Menlo Park, CA 94205. Publications include a directory of members, *AI Magazine*, and proceedings of yearly conferences. The magazine offers a mix of theoretical, practical, and general interest reading.

Computer Society of the Institute of Electrical and Electronic Engineers (IEEE), 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. Publications include *COMPUTER*, *IEEE Software*, and *IEEE EXPERT*. IEEE EXPERT magazine is devoted to articles on applied AI; articles on AI also appear in the other magazines.

1.9 QUICK BRIEFING ON EXPERT SYSTEMS

What is an Expert System?

- Perspectives
- Definitions
- Component parts
- Examples

"Expert Systems" Comprise a Branch of "Artificial Intelligence"

- AI is a collection of cutting-edge, integrated concepts and methods from a variety of fields:

  -- Decision Science
  -- Computer Science
  -- Linguistics
  -- Psychology
  -- Neurology
  -- Mechanical and Electrical Engineering

  and many more...

- AI is what enables computers to perform tasks that would normally require an experienced or trained person

"Expert Systems" Are A Practical Application of Artificial Intelligence

- Expert systems have proven to be practical in areas such as:
  - -- Medical diagnosis;
  - -- Seismic analysis and oil prospecting;
  - -- Chemical analysis;
  - -- Preventive maintenance;
  - -- Crisis response;
  - -- Sales and negotiations advice;
  - -- Games; and
  - -- Engine fault diagnosis.
- The expert systems arena is one of the most commercially advanced branches of AI.

Some Classic Definitions of "Expert Systems"

"A computer system that ***emulates human expertise*** by making deductions from given information using the rules of logical inference."

"A system whose goal it is to perform convincingly as an ***advisory consultant***, exhibiting ***human expertise in a given domain*** with ***self-explanation of reasoning*** on demand."

"A computer program that has built into it the knowledge and capability that will allow it to ***operate at the expert's level.*** It will usually be able to explain the ***lines of reasoning*** that led to their decisions."

Conventional vs. Expert Systems

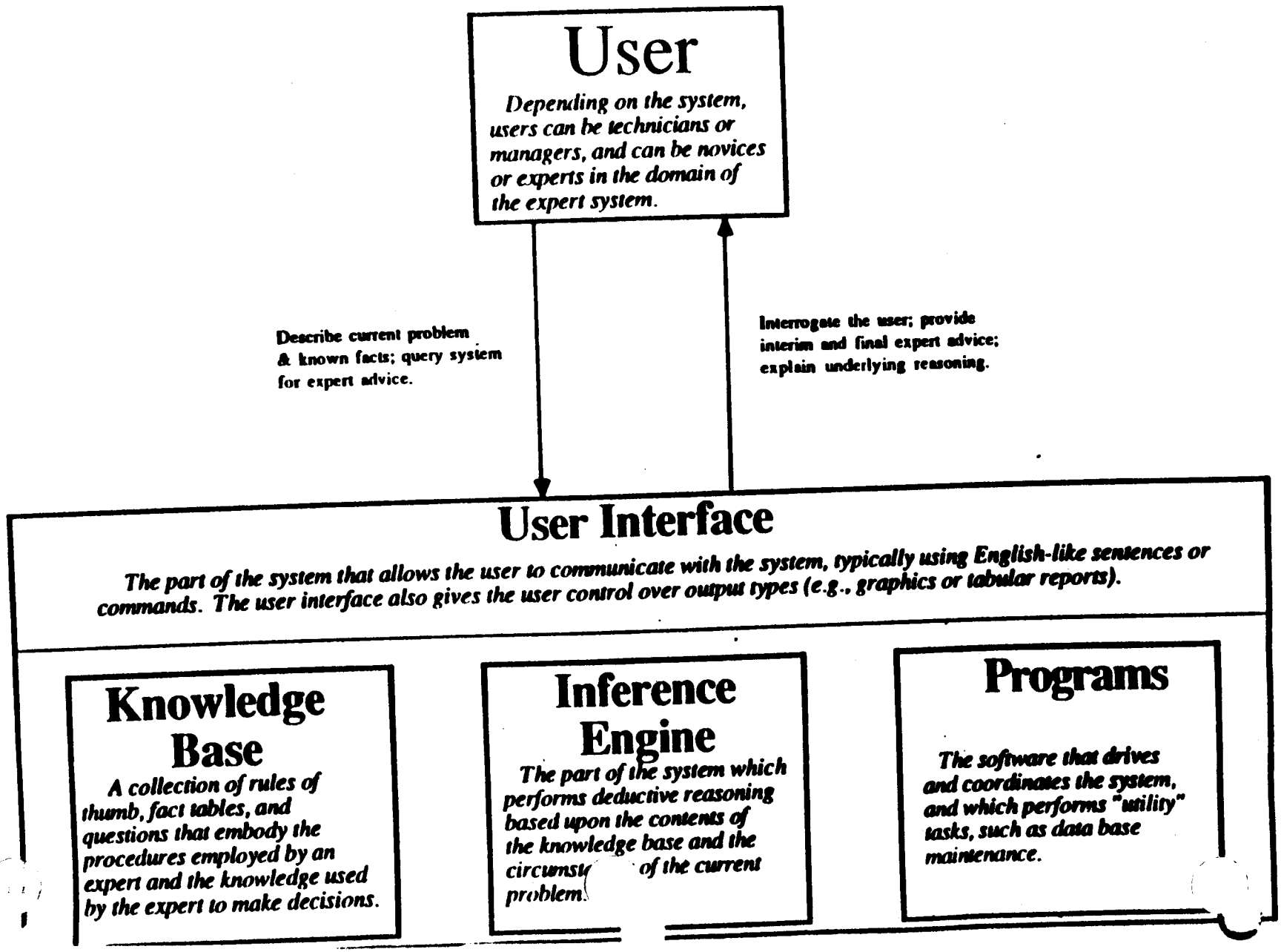
Conventional

- Unambiguous data
- Strict formats
- No mid-run explanations
- Numeric processing
- Major effort to modify

Expert

- Uncertain/incomplete data
- Rules of thumb
- Mid-run explanations easy
- Symbolic processing
- Easier to modify

Key Components of an Expert System



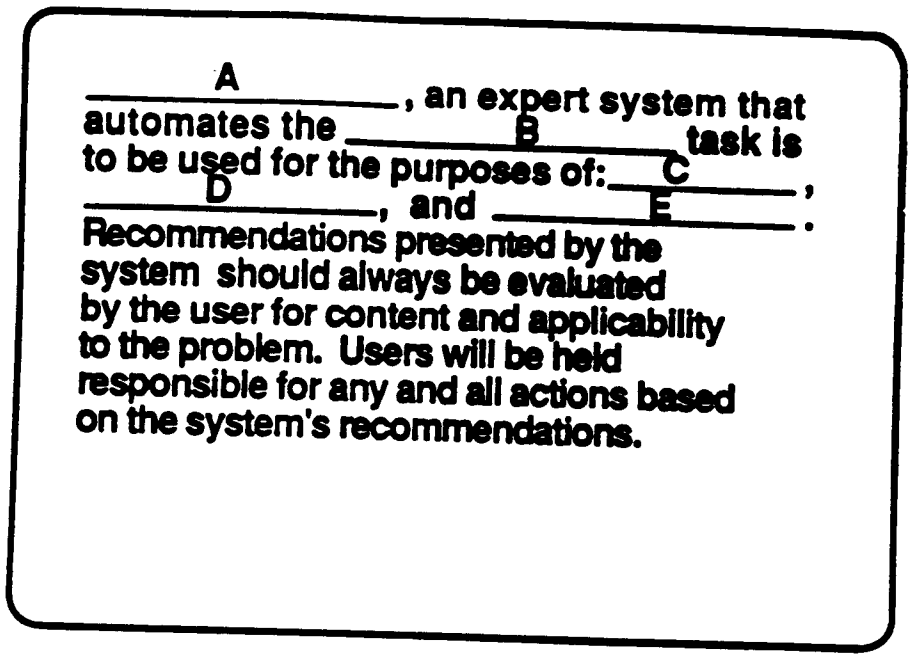
The Knowledge Engineering Process

- Information is extracted from experts
- Information (i.e., rules, facts, and procedures) is arranged in a knowledge base
- Information conflicts and voids are identified and resolved

Hardware Environment vs. Expert System Application Characteristics

Environment	Characteristics
• Microcomputer	• Rapid/Proof-of-Concept Prototypes • Script-based • Rule-based with limited fact tables • Limited requirement for access to external data files
• Minicomputer -or- AI Workstation	• > 1000 rules • Processing speed a consideration • Resource-intensive processing • Efficiency of operation required (i.e., AI architecture, or garbage collection)
• Mainframe	• Multi-user access • >1000 rules • Processing speed a consideration • Access Required to Large Data Bases • Resource-intensive processing

(



Key

A - name of expert system
B - title for task automated by expert system
C, D, E - activities automated by the expert system

Figure 2.1
Example of an Expert System Disclaimer

CHAPTER 2
IMPLICATIONS OF USING EXPERT SYSTEMS AT EPA

2.0 PURPOSE OF THIS CHAPTER

This chapter discusses some policy issues for expert systems.

2.1 EXPERT SYSTEMS FOR POLICY INTERPRETATION

Expert systems for policy interpretation are not cost-effective for OSWER because of the dynamic nature of policy. The maintenance costs of expert systems that address policy can be significant due to changing directives from Congress and regulatory policies. Projects that address policy should have sufficiently short payback to offset their risk of obsolescence. Because of these difficulties, expert systems may be used to cite guidance and regulations but policy interpretation should be avoided. This does not impose restrictions on text retrieval systems -- such as hypertext and smart indexing -- but interpreting the policy is to be left to human decision makers.

2.2 LEGAL ISSUES

Legal issues for expert systems involve all matters of the law, including liability. Legal issues include licensing and distribution rights to the software, copyright infringement risks for software and documentation, and access to information used in the expert systems.

2.2.1 Licensing Agreements

All OSWER expert systems will adhere to the terms of negotiated licensing agreements set up with software vendors.

2.2.2 Confidential Information

Expert systems and their contents are open to public inspection. Therefore expert systems should not contain confidential business information (CBI) or enforcement information, nor should they contain data covered by the Privacy Act. Information used in the expert systems will be made available to the public in accordance with the Freedom of Information Act.

2.2.3 Liability

Liability in the realm of expert systems has received much attention from computer scientists, but has not been tested in the courts. Some software liability cases have been tried -- such as the LOTUS 1-2-3 spreadsheet error suit -- but the field is relatively open to interpretation.

2.2.3.1 System Profile

Expert systems appear to be especially vulnerable to liability issues because of their advisory nature in judgmental matters and their high profile. The high profile can be addressed by classifying the use of an expert system as an assistant or advisor. This sets the user's expectations to a more reasonable level. If people believe that they are receiving accurate, "expert" information, then they may act on it without the proper skepticism.

2.2.3.2 User Registration

Another means of reducing liability is to control the people who use the expert system by registering, or characterizing and qualifying them. Registering the users of expert system is discussed in Chapter 7. Characterizing and qualifying users involves three steps.

First, developers must assess the degree of computer literacy and competency in the subject area of the intended users and build the system accordingly. Second, the users must be made aware of the level of complexity and scope of the expert system via disclaimers and documentation. Finally, initial and on-going training is essential, especially for first-time users of the expert system. User qualification can be enforced by requiring that the user specify a registration number for the expert system when requesting user or technical support.

2.2.3.3 Disclaimers

The proper use of disclaimers (Fig. 2.1) will also reduce the risk of liability for expert systems. Disclaimers should include information on:

- The characteristics of the intended user of the expert system;
- The scope and applicability of the system;
- Assumptions made in the development process; and
- Intended uses for the system.

It should be explicitly stated that using the expert system for purposes other than those intended will produce unpredictable results, and that neither EPA nor its contractors are responsible for the consequences resulting from intentional abuse or unauthorized use of the system..

2.2.3.4 Liability Scenarios

For now, EPA should consider expert systems to be tools, with the decision-maker retaining the responsibility for any liability resulting from the decision. EPA managers are not divested of liability, but can reduce it by emphasizing the advisory nature of expert systems. Some researchers suggest that expert systems are simply automated information retrieval systems and should be held no more liable than publishers of books. Another point of view suggests that expert systems be held liable as independent experts, and thus are fully responsible for all recommendations. In other words, the expert system is just a tool that people may use at their discretion, but may in no way hold responsible for the outcome.

Finally, there is the possibility that people may be held liable for not using an expert system. If a mistake is made that could have been avoided by using an expert system, the decision-maker might be found at fault. Under each of these liability scenarios it is important to think about the extent of liability for each person involved -- including the expert, the developers, the users, and the maintenance staff.

2.2.3.5 Private Industry

Developers of EPA systems should consider the liability implications of EPA-sanctioned software being used by private industry. Expert systems developed by EPA can have broad ranging effects in the hands of unauthorized users. Thus EPA should implement a user qualification system and a user registration system (see CHAPTER 7) to mitigate these effects. Additionally, EPA expert systems may wind up in the hands of the regulated community as a result of a Freedom of Information Act (FOIA) request.

2.2.3.6 Distribution

Because of the potentially extensive user base of each expert system, it is necessary to consider the implications of their use on Regions, states, and industry. Three mechanisms for distributing expert systems are:

- User registration,
- Mailing lists, and
- Open dissemination.

The extent of risk for misuse of an expert system increases the more freely it is distributed. The type of distribution is left to the project manager's judgment based on the sensitivity of the information contained in the expert system.

2.3 ORGANIZATION'S CULTURE

The initial approval to proceed with an expert system project and its final acceptance depend upon an organization's culture. In addition, the culture will in some way be affected by expert systems technology, if it is incorporated into the operation of the organization.

2.3.1 Level of Acceptance

The level of acceptance of new technology depends on an organization's responsibility and risk tolerance. Conservative organizations that operate in a well-defined environment are often reluctant to try advanced technical information system solutions. Other organizations view advanced technology as an opportunity to increase productivity and capabilities, and thus welcome expert systems.

2.3.2 Developer Initiative

Developer initiative is heavily influenced by an organization's culture. Organizations that encourage individual initiative will stimulate more activity in new technologies, and are therefore more likely to develop grass-roots expert system ideas and projects.

2.3.3 User Acceptance

User acceptance also depends on the nature of the organization. The successful introduction of new technology is often based on how the users of that technology are received. In other words, if users are encouraged and rewarded for their efforts, the new technology is likely to succeed.

Another factor that affects user acceptance is management commitment. Positive reinforcement from management encourages user acceptance. Involving the users in the early stages of the expert system's development increases their feeling of ownership and thus promotes user acceptance. If the users are indirectly discouraged or penalized for aborted attempts, the new technology often remains unused. This is especially critical during the prototyping stages.

2.3.4 Prototyping

Organizations that embrace new technology should understand the exploratory nature of rapid prototyping used in developing expert systems. Rapid prototyping provides the flexibility to make false starts within a project (see Section 1.7). Because prototypes are not all-or-nothing efforts, they may be discarded in favor of a new approach as new information is discovered. The success of expert systems depends on organizations being tolerant of the exploratory process that might not have immediate benefits.

2.3.5 Effects of Expert Systems

Changes in the operation of an organization lead to changes within the organization's culture. Expert systems can cause changes in two ways. First, in building an expert system to automate a task, the task itself comes under scrutiny. This in itself often causes changes in the procedure. Second, expert systems affect the information task itself by changing the user's input routine, the time involved in finishing the task, the answer and justification received, and potentially in other quantitative and qualitative ways.

Two other facets of expert system development that are influenced by an organization's culture are effects on expert system development staff and ramifications of early expert system projects. Finally, the organization's culture determines the amount of emphasis that is placed on early expert system initiatives. Further efforts in expert systems may hinge upon the success of earlier ones.

2.4 CROSS CUTTING CONSIDERATIONS

Several topics of particular importance are addressed in multiple phases of the system life cycle. These topics include:

- Project Management Plan;
- Project Participation;
- Project Reviews and Quality Assurance;
- Project Approvals;
- Methodologies and Tools;
- Benefit-Cost Analyses; and
- Knowledge Management.

This Section briefly describes each topic from a cross cutting, life cycle wide perspective. Specific activities relating to each of these topics are presented in the remaining chapters of this practice paper.

2.4.1 Project Management Plan

The Project Management Plan is the fundamental document for planning and managing the system life cycle, and is mandatory for every project. It is first developed in the Initiation Phase and is updated, expanded, and refined continually throughout the life cycle. Refer to the *OSWER Project Management Practice Paper* for additional information.

Several important implications of expert systems for the Project Management Plan are noted below:

- Explicit determination of the relevance of the application area and type must be documented in the Project Management Plan.
- Although prototyping can be an iterative, repetitive process, justification for each successive prototype must be documented in the Project Management Plan. The "lessons learned" in each iteration should be spelled out.
- Any modification to the standard life cycle stages and phases to accommodate prototyping or other aspects of expert system development must be documented in the Project Management Plan.
- Knowledge base testing, validation, and maintenance plans must also be spelled out in the initial plan and refined over time.

2.4.2 Project Participation

Information management problems and systems to address them require the participation of organizations and individuals with a diverse set of experience and skills. These range from an in-depth understanding of pertinent agency programs to expertise in specific information management technologies. Successful projects require that certain important roles be specifically assigned to organizations and individuals.

Expert system development efforts will require the roles common to all system development projects. These are OSWER Program Management, OSWER Program Execution, Project Management, Project Execution, Quality Assurance, and Procurement. Within Project Execution there are the following specialized roles.

- [Domain Expert] - the expert provides the information or expertise for the expert system. The expert acts as the functional expert; providing expertise, defining the boundaries of the domain, and providing validation for the expert system. The expert's qualification is to have in-depth knowledge of a functional area.

- [Knowledge Engineer] - the knowledge engineer elicits the information or expertise from the expert. The knowledge engineer's role in an expert system project is to gather the information or expertise of the expert and to codify that information using expert system techniques. The knowledge engineer's qualifications are to be familiar with expert system development tools and methodologies.

- User - the user is a person who requires the functional expertise of the expert on a regular basis. This individual, over time, will become a functional expert. The user's role in an expert system project is to use the system to perform tasks that would normally require input from an expert. The user's qualifications are to have a willingness to work with new systems and to understand the functional area to the degree that he or she can identify when the expert system is providing incorrect recommendations.

2.4.3 Project Reviews and Quality Assurance

Independent review of the products of the system life cycle is performed to ensure that the project team is proceeding in an appropriate direction to effectively solve the information management problem. The reviews address programmatic issues, technical considerations, and project management. The reviews provide feedback to the project team as well as advice to the individuals required to approve the project. Project reviews

and other quality assurance activities are performed throughout the life cycle.

Several important aspects of expert system project reviews and quality assurance are noted below:

- The reviews to be conducted and other planned quality assurance activities are documented in the Project Management Plan.
- Quality assurance applies to all aspects of the project, and quality assurance is a continual part of the project effort.
- Formal reviews are structured to ensure a level of review commensurate with the nature and scope of the information management problem and potential solution.
- The results of reviews are included in each decision paper developed at the end of each stage of the life cycle management process.
- Knowledge base testing and validation will require separate verification of data (facts) and logic (rules)
- Detailed test cases and review and approval of the system by the expert will be required.
- Problems particular to knowledge bases will have to be isolated and corrected such as subsumed rules, contradictory rules, unreachable goal states.

2.4.4 Project Approvals

Formal approvals are provided throughout the system life cycle to ensure that OSWER management supports the project and is in agreement with the chosen project direction. These approvals are provided at a level commensurate with the nature and scope of the system. Review and approval thresholds established for conventional systems may not be appropriate for expert systems. The following is a review structure that considers impact, risk, level of interest, usage distribution, development effort, and system type follows:

- Impact can be measured in several different ways. One is to estimate savings in terms of some combination of professional labor hours saved, support labor hours saved, computer time saved, or reduced contractor assistance.
- Risk is a measure of a system's susceptibility to incorrect or partially incorrect results and the probable significance of such results. The measurement of risk

takes into account such factors as the cost of incorrect or partially incorrect results, a cost-benefit analysis of the system, and a risk analysis of the "regret factor."

- Level of interest in a particular system is a measure of its visibility. Exposure of the system within the agency, the government as a whole, or the public eye, will influence the degree of review and approval required for the system.

- Usage distribution will affect review and approval requirements. One possible distribution categorization is expert systems developed for individual use systems, work unit distribution, Agency-wide distribution, and distribution beyond the Agency. Each level of distribution would require increasing levels of review and approval.

- Development effort is a measure of the amount of resources invested in the development and maintenance of an expert system. Investments should be calculated in terms of dollars and labor hours (usually full time equivalents) required to develop the expert system.

- System type or application domain of a given expert system will also affect its level of review and approval. Some domains such as hazardous waste control will demand near perfection and resulting intensive review. Other domains such automated forms processing may require much less review.

2.4.5 Methodologies and Tools

Expert systems projects will clearly use different methodologies and tools than those used in conventional systems development. Several important expert system considerations in the use of methodologies and tools are noted below:

- While the *LCM Guidance* does not mandate the use any specific methods or automated tools, it does require that choices be made explicitly. Proper selection of application types and areas facilitate identifying required expert system shell characteristics, which in turn lead to specific products.

- Successive prototypes must be justified individually and must be documented in the Project Management Plan.

- Methodologies and tools should be considered from a full system life cycle perspective.

- The methodologies and tools for each phase or stage are selected no later than the end of the preceding phase or stage.

2.4.6 Benefit-Cost Analyses

The important decisions of the system life cycle and the prescribed management approvals often hinge on two key questions: "How much will it cost?", and "Are the benefits worth the cost?" Benefit-cost analyses are particularly important in expert system development because of the unique hardware, software, and developer skills that may be involved. However, quantifying these measures for expert systems can be difficult, so alternate measurement techniques should be considered. These may include:

- Lost opportunity cost analysis,
- Relative worth methods,
- Quantitative bounding, and
- Qualitative bounding.

2.4.7 Knowledge Management

Knowledge management is the process of formally controlling the initial development of a knowledge base and all subsequent changes or additions to it. It ensures that the integrity of an expert system is maintained throughout its life cycle. Knowledge management is closely related to configuration management. Refer to the *OSWER Configuration Management Practice Paper* for a more complete description.

There are several advantages of applied knowledge management over less structured knowledge base development. As the agency develops multiple expert systems over a period of time, opportunities will present themselves to gain leverage from previous efforts. Well-structured knowledge bases will permit the development of reusable modules of knowledge stored in a "knowledge library." A reusable "knowledge dictionary" will enhance expert system development by reducing the time and cost required to specify commonly used knowledge base elements.

As OSWER increases its expert system development activities, it should make a deliberate effort to seek out other systems currently under development in order to identify opportunities to share knowledge. If multiple applications can access common knowledge bases, they can be cost-justified more easily and accurately.

Linking modules together to form a knowledge base may impact system performance adversely or positively. On the one hand, the control structure required to interface the different modules will consume some portion of the processing capabilities of the system. On the other, breaking a knowledge base into modules may permit loading and running only those modules needed for the current consultation.

Knowledge management is an ongoing process because expertise is constantly evolving and changing. Proposed changes and additions to the knowledge base must undergo formal review before they are made to ensure the integrity of the system. Some requested changes may be of such significance that they must be addressed through a separate iteration of the life cycle. Requests which will have the effect of changing the types of data and/or knowledge to be processed by the system generally should be handled in this manner.

2.5 RESOURCE REQUIREMENTS

Resource requirement considerations comprise a significant portion of the planning process for expert system development. Particular attention should be paid to scarce resources such as an expert's time. Expected requirements for resources in each phase of the life cycle are discussed below.

2.5.1 Experts

The single most important resource in the development of an expert system is the time of the domain experts. Their time is also a scarce resource because of the demands that placed on them by their organization. The commitment of an expert's time is a plus in terms of management support, but it is also a risk in terms of not wasting this business resource on fruitless efforts.

2.5.1.1 Initiation Phase

The experts are needed as a resource in the Initiation Phase to assist in determining the scope of the system because they know the problem area and solution requirements best.

2.5.1.2 Concept Phase

Experts are needed in the Concept Phase to determine the initial expert system functions and features. In addition, the experts will develop a preliminary overview of the problem that allows the developer to build a good proof-of-concept prototype. Involvement of the experts is extensive in the Concept Phase.

2.5.1.3 Definition and Design Phase

The experts should be prepared for a significant time investment in the Definition and Design Phase. During this phase the experts must explain the problem to knowledge engineers and to management. The experts are also crucial for the development of the demonstration prototype of the expert system built in this phase.

2.5.1.4 Development Stage

When determining the amount of time needed from the experts, remember the experts are going to be needed more in the initial startup of development where most of the knowledge acquisition takes place. As the development progresses, the experts will need to be consulted on a regular basis. The experts will also be needed during testing, verification and validation.

2.5.1.5 Implementation Stage

During implementation of the system, experts will be needed to a lesser extent than during the Development Stage. They will be needed mainly to resolve any problems that arise during the beta test. It may be useful to find another expert in the field who was not involved in the development of the system to serve as a beta tester. The additional expert will be able to test the system more rigorously than other beta users, and could assist in a final validation of the system.

2.5.1.6 Operation Phase

Experts will be needed to assist in modifying the knowledge base in the Evaluation Stage. They will also be helpful in determining the degree to which the system continues to address the information management problem. The size of the system and the volatility of its problem domain will dictate how much time will be required from the experts.

Experts may also be helpful in determining what portions (if any) of the terminated system to retain in the Archive Stage.

2.5.2 Information about the Problem

Information about the problem will be required to verify that the problem exists and to justify the system development effort to solve it. The information will be gathered in the Initiation Phase and will be used to document the existence of the problem and, when available, describe solutions to similar problems.

2.5.3 Knowledge Engineers

Knowledge engineers perform three major functions in the expert system development life cycle. First, they help structure the problem, perform feasibility studies, and design the expert system. Second, knowledge engineers extract knowledge about the problem solving process from the experts. Finally, they are responsible for knowledge management in the expert system.

Because of their important role in the development process, knowledge engineers are extensively involved in the expert system project from the Concept Phase on.

2.5.3.1 Concept Phase

Knowledge engineers perform a significant portion of their work in the Concept Phase with responsibilities including:

- Selecting the knowledge representation and control structure,
- Developing the Knowledge Management Plan and System Test Plan,
- Helping determine the functional requirements of the expert system,
- Assisting in performing a feasibility assessment for the expert system approach, and
- Building the Proof-of-Concept prototype.

The amount of time spent by the knowledge engineers will vary with the size and complexity of the application.

2.5.3.2 Definition and Design Phase

The number of knowledge engineers is dependent upon the estimated size and complexity of the problem. It is recommended that two knowledge engineers interact with the experts in order to provide different points of view in the knowledge acquisition process. During design, the role of the knowledge engineers is to assist in identifying potential knowledge sources and to help the experts to explore possible avenues in the problem domain. However, it is important for the knowledge engineers to help the expert scope the problem domain down to a reasonable size in order to design a technically feasible expert system.

2.5.3.3 Development Stage

The knowledge engineers will be needed throughout the Development Stage. The project schedule hinges on their level of commitment. Initially, the knowledge engineers will be involved extensively in documenting the expert's knowledge. The knowledge engineers also provide support to the programmers; in some cases, the knowledge engineers will perform both knowledge engineering and programming. Finally, the knowledge engineers will interact with the experts, programmers (if used) and management during the testing and validation phases of development.

2.5.3.4 Implementation Stage

Knowledge engineers will play a much lesser role during this stage than the Development Stage. They will be needed mainly for contacting the experts if any problems arise during beta testing. They will also be involved in analyzing the comments and suggestions of the beta users, and making any necessary revisions to the system. Knowledge engineers may also be involved in the training process and will provide expertise for the expert system documentation.

2.5.4 Programmers

Programmers are responsible for the development of software associated with the expert system. This includes placing the knowledge acquired from the experts into the development environment, writing interfaces, developing support programs, and integrating the expert system into the existing information processing architecture. In some projects, the knowledge engineers may serve as the programmers.

2.5.4.1 Concept Phase

Programmers who have experience with expert systems are necessary to help determine the testing methodology in the Concept Phase. They also assist the knowledge engineers build the Proof-of-Concept prototype.

2.5.4.2 Definition and Design Phase

It is the responsibility of the programmers to incorporate the data extracted during knowledge acquisition into the specified development environment. They can be used to assist in the requirements definition process and as a technical resource for programming issues that need to be addressed in the design. They are also useful for building the Demonstration Prototype during this phase.

2.5.4.3 Development Stage

In the Development Stage, the programmers are responsible for helping codify the experts' knowledge and for writing support programs. They will be the key people in building the knowledge base, developing the interfaces, and responding to the users suggestions on how to improve the expert system. Their services will continue through testing and validation where they will be responsible for any changes necessary before the Implementation Stage.

2.5.4.4 Implementation Stage

The programmers will be involved to a lesser extent in this stage than in the Development Stage. They will be involved in making any necessary revisions to the system following the beta test, and may provide some input for the documentation on the system. Programmers could also be involved in technical support to the beta users, if there are any problems at the program level or problems with hardware and software compatibility.

2.5.5 Data Collection

Effective data collection facilitates the development of expert systems. Most of the data collection is performed in the early stages of the life cycle.

2.5.5.1 Concept Phase

Initial data collection functions are limited to those necessary in determining the conceptual model's features and inputs, user training, system benefits and costs, end users, and system scope. Potential sources of information include other expert systems that are already built, in the Initiation Phase, or under development. Resources are mostly those of the knowledge engineer's time.

2.5.5.2 Definition and Design Stage

Further data collection required during the Definition and Design Phase involves information about the problem domain, the development environment, and the targeted delivery environment. To properly define and scope the problem domain, possible data sources should be identified and realistic objectives should be established.

In the selection of the development and delivery environments, much information is required in order to make a knowledgeable decision. For example, in order to assess the cost of the system, the hardware requirements should be determined for both development and delivery environments.

2.5.5.3 Development Stage

In the Development Stage, data will be collected on user feedback, management feedback, and the development process. Also, any data in the form of spreadsheet and database will need to be documented in the development plan. Use the data management plan as a guide in the data collection process.

2.5.6 Licensing Fees

It is important to consider licensing fees for both the development and the delivery environments when estimating the resource requirements for the expert systems. The licensing fees constrain the number of copies of software that may be used in developing the expert system, and also the number of copies of the expert system that can be distributed.

2.5.6.1 Concept Phase

Planning for software runtime licensing fees is necessary in this phase because both the costs to build and to field the expert system are based on the number and location of the users.

2.5.6.2 Implementation Stage

As the system is nearing distribution to field users, a staff member will be needed to check all licensing and run-time fees with the company who distributes the shell. All fees must be paid, contracts and agreements must be activated, and any other legal matters must be addressed at this time.

2.5.7 Testing

The testing methodology needs to be set in the Concept Phase. Both conventional software and expert system testing methodologies should be considered. Testing is done by the development staff if performed during the Development Stage. The testing includes validation of the data, knowledge base, and logic flow.

During the Implementation Stage, the system will be tested in the field by potential end users. Comments received from those testing the product will be incorporated into the final revisions. This testing will help to ensure user acceptance of the system.

2.5.8 Maintenance

Maintenance plans should be started in the Concept Phase. Questions such as to who will be responsible for maintenance, when it will be performed, how volatile and dynamic the problem domain is, and what role the users will play need to be addressed. A staff member or contractor will be needed to continue to maintain the expert system, as well as provide technical support to the beta users during the Implementation Stage.

Maintenance is necessary in the Operation Phase to ensure that any previously undetected errors can be corrected. Maintenance also provides a means to take advantage of hardware upgrades, new releases of system software, and applications packages that enhance the system's performance.

2.5.9 Hardware

The development hardware will be needed from the Concept Phase through the Development Stage. For more information on selecting suitable hardware refer to Section 5.4.1.3. The beta users will provide their own hardware during the Implementation Stage.

Changes to the expert system, prompted by advances in hardware and software, may be necessary during the Operation Phase. Changing the hardware platform of an existing system should be avoided if at all possible, but if the existing platform is no longer supported or becomes obsolete, it may be unavoidable. Such a change will likely warrant a separate iteration of the system life cycle.

2.5.10 Software

By the end of the Concept Phase an appropriate development vehicle, including the expert system shell, environment, and language, has been selected. The key issues here are how to encode the knowledge and how to integrate the software packages to expedite development. Topics to consider are:

- Interfaces between any external and/or utility programs. Also, any links between quantitative techniques and the expert system should be addressed.
- Identify and use existing software development aides such as intelligent editors, debuggers and source code managers.

Copies of the run-time version of the system will be made during the Implementation Stage and sent to the beta users. Over the span of the Production Stage, the knowledge base will likely be modified one or more times. These modifications may entail interfacing the expert system with newly developed systems or upgrading the system's shell. All software and data that are unique and essential to the system's operation should be archived for possible future use.

CHAPTER 3
INITIATION PHASE

3.0 INTRODUCTION

Initiation is the first of five major phases in the OSWER system life cycle. The other phases are Concept, Definition and Design, Development and Implementation, and Operation. This phase provides the first formal description of the information management problem and secures the resources needed to examine the problem and potential solutions. The most significant activities of the Initiation Phase include:

- Confirming the existence of a problem, providing additional guidance where appropriate, and approving (or rejecting) the commitment of resources to begin the Concept Phase. This activity is performed by program management.

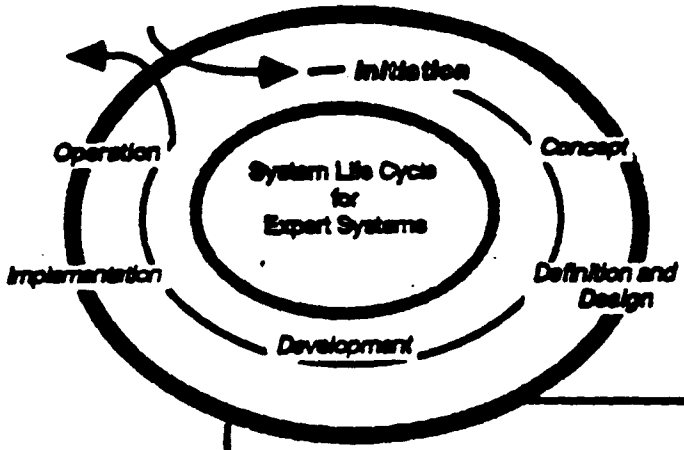
- Preparing an Initiation Decision Paper that identifies and describes the information management problem and brings it to the attention of OSWER program management. This activity is performed by one or more program organizations.

- Preparing a Project Management Plan to describe the initial approach for managing and conducting this phase and future actions under the remainder of the life cycle. At this point, the Project Management Plan provides detail for the Concept Phase; it is revised and enhanced in later phases.

Clearly identifying and describing the information management problem is critical to the successful development of an appropriate solution. How the problem is defined during Initiation will shape the analyses and decisions of the subsequent life cycle phases.

> During the Initiation Phase, there should be no assumption that the solution will necessarily be an expert system!

Even in cases where the system developers have preconceived notions that an expert system is the preferred solution, the analysis in the Initiation Phase should not be intentionally slanted to use expert systems. Instead, the primary goal of this phase is to describe the information management problem objectively, independent of potential technological solutions.



Objectives

- To describe the problem in clear, technology-independent terms upon which organizations can agree
- To determine whether staff or other resources will be devoted to defining and evaluating alternative ways to respond to the identified problem in the Concept Phase

Decisions

- Project approach decisions including:
  - who will manage the project
  - who will participate in development
  - what reviews are necessary (and by whom)
  - what approvals are necessary (and by whom)
- Execution decisions including:
  - what is the problem
  - what organizations are involved
  - what are the sources of data, the scope of the solution, and the information requirements such as report generation

New Products

- Initiation Decision Paper
- Project Management Plan

Figure 3.1

Initiation Phase
Objectives, Decisions, and Products

3.1 OBJECTIVES

Figure 3.1 graphically illustrates the objectives associated with the Initiation Phase.

The primary objective of the Initiation Phase is to describe the problem in clear, technology-independent terms upon which all pertinent organizations can agree. If an expert system may be the technology chosen to resolve the problem, then the project manager and software developer should consider the expert system application success factors listed in Section 3.3 before completing this phase.

The second objective for the Initiation Phase is to determine whether staff or other resources will be devoted to defining and evaluating alternative ways to respond to the identified problem in the Concept Phase. Committing resources beyond the Concept Phase is premature at this point. The use of rapid prototyping as a tool to work through these issues is a feasible approach at this stage and in the Concept Phase.

3.2 DECISIONS

There are three types of decisions made in the Initiation Phase: project approach, execution, and continuation. Figure 3.1 graphically illustrates these decisions.

3.2.1 Project Approach

The project approach decisions address the organization of the project and the participants in the project activities such as system acceptance testing, reviews, and approvals.

Decisions are focused on project management and controls, establishing:

- Who will manage the project,
- Who will participate in development,
- What reviews are necessary (and by whom), and
- What approvals are necessary (and by whom).

3.2.2 Execution

The execution decisions address the scope and specific features of the system. These decisions address programmatic, technical, and system support related issues.

Programmatic issues include deciding what the problem is, whether an organization is currently responsible for solving the problem, and which organization(s) should receive the proposed automated solution. In resolving

these issues the emphasis in the Initiation Phase should not be on a special technology (e.g., optical disk technology, expert systems, etc.) but should be technology-independent.

3.23 Continuation

The continuation decision confirms that a defined information management problem exists and is significant enough to warrant further investigation. The decision confirms that the information management problem is beyond the capabilities of existing systems and that developing a new system has merit. The decision made here also has to do with resources, and this is an assessment of return on investment, and that is that the investment of people and dollars will produce results which are desired by the organization in terms of productivity.

Technical issues include determining sources of data, the scope of the solution, and the information requirements such as report generation. When there is a mix of conventional systems and expert systems -- called hybrids or embedded systems, new information needs may be associated with the conventional systems being considered; in this case, the Initiation Decision Paper should identify the new information needs. A systems support issue is to determine who will be responsible for the system once it is fully operational.

3.3 SUCCESS FACTORS

Several factors that can impede success if not considered in the Initiation Phase are described below. This Section focuses on perceptions of what expert systems can and cannot do, and it offers solutions or alternatives.

3.3.1 System Development Bias

Because expert systems are a relatively new technology, individuals and groups involved in initiating a new system may have little or no familiarity with them. They may even have a negative impression of their capabilities. On the other hand, some parties may tend to find an expert system solution to every problem, whether it is appropriate or not. Before undertaking an expert system project, it is important to determine that the problem has not already been solved by other types of conventional programming such as modeling, decision support, databases, or linear programming.

3.3.2 Scoping the Problem

Expert system development projects must be reasonably scoped from initiation onward. Failure to do so can lead to developing a solution to the wrong problem or tackling overly complex or nebulous problems.

3.3.3 The Problem is Well Suited to ES Technology

The key characteristic of an application that will help the project manager or software developer determine whether expert system technology should be used is the type of problem. If the problem is purely algorithmic or procedural in nature, then it can be addressed by conventional technologies more efficiently than by expert systems. If the type of problem requires symbolic reasoning (as described in CHAPTER 1, Expert Systems Defined, then the problem may be suitable for expert systems technology.

3.3.4 An Expert is Available

Expert systems are developed by taking the specific knowledge of an established expert and putting it into a system. Some systems may incorporate the knowledge of more than one expert, while others reflect the knowledge and strategies of a single individual. Finding the right expert(s) is a key step in building an expert system. If no true expert exists, then the problem may be too nebulous and ill-defined to be effectively addressed by an expert system.

3.3.5 The Problem Domain has Bounds

Once an expert is identified, the project manager or software developer should consider whether the set of solutions to the problem has boundaries or whether there are infinite solutions. The problem cannot have an infinite set of solutions. The solutions that will be considered by the system must be determined in advance. When the expert system attempts to wor with information from near the periphery of the problem domain, it may yield unpredictable results. The problem domain must be readily defined using empirical knowledge such as facts, rules, or algorithms. It should not require common sense (i.e., knowledge everyone takes for granted, but few can articulate) or sensory data (i.e., vision or sense of smell).

3.3.6 An Expert Can Solve the Problem in Less than a Week

The project manager or software developer should also consider the complexity of the tasks that are to be automated with an expert system. The tasks should neither be too difficult nor too trivial for a human expert. A task requiring more than a week to solve without computer support is almost certainly too large to be thoroughly delineated using rules. However, if the problem can be parsed into subproblems, it may be manageable. On the other hand, a task requiring only a few moments to solve might be automated more efficiently using conventional technologies.

3.3.7 A Significant Return on Investment is Anticipated

Since expert system development requires a significant investment in terms of people and money, the expected return on that investment must be well understood along with the means of measuring the return. Can the relative

success of the expert system's performance be assessed? The Initiation Phase should address this issue before moving to the Concept Phase.

3.4 PRODUCTS

Many products are produced and/or updated in the course of the system life cycle. The Initiation Phase products are described below.

3.4.1 Initiation Decision Paper

The Initiation Decision Paper describes the information management problem and justifies undertaking the next phase of the life cycle. Whether an expert system is being considered as the solution or not, the Initiation Decision Paper should be written in technology - independent language and should focus primarily on the scope and magnitude of the problem at hand. It may highlight the attributes of the problem that are suggestive of a possible expert system solution (refer to Section 3.3, "Success Factors"). The key parts of the Initiation Decision Paper that will affect expert system projects are:

- Mission areas addressed: boundaries and domain for the problem must be clearly established.
- Description of the problem: conventional ways of describing information management problems such as flow charts and data flow diagrams may be only partially useful for an expert system-type problem. Different decision [representation] techniques include decision trees, rules, or just plain English explanations may be preferable.
- Overall project approach: methodologies and tools may be different for expert systems. These issues are discussed as cross-cutting considerations in Chapter 9.

Prototyping may be used as a tool to develop the analytic basis for the decision paper.

3.4.2 Project Management Plan

Developing the preliminary Project Management Plan, the second product associated with the Initiation Phase, is an important step because it results in the collection of the following information:

- Preliminary life cycle cost estimate,
- Detailed estimate for the Concept Phase,

- Results of threshold analysis of appropriate levels of review and approval,
- Methodologies and selection of tools to be used in the Concept Phase, including prototyping,
- Results of benefit/cost benefit analyses,
- A list of experts who will provide the knowledge base of the expert system, and
- A list of users who will ultimately be responsible for the success or failure of the system.

Techniques for justifying expert systems are in many ways alike to those used to justify other information management systems. Some of the techniques used to justify expert systems are described below.

- Benefit Analysis - This approach takes advantage of tangible and quantifiable expert system benefits and attempts to quantify the qualitative benefits. Depending upon the measures of success of the expert system defined early in the development cycle, there may be significant economical advantages to building an expert system. Managers need to be aware of the applicability of this approach to justifying expert system projects and the measurability of benefits such as "improved productivity" and "improved accuracy" such as what mistakes have cost in the past. Also, how would the resources be applied if the expert system is not deployed?
- Cost Savings Analysis - A second approach to justifying an expert systems involves the cost savings. While expert system development may seem expensive, the cost of remaining with the current system over time may prove to be more expensive.

Depending on the magnitude and type of problem being considered, several expert system methodologies and tools may be used in later phases. These will be identified in the Concept and Definition and Design Phases. Expert system-specific issues include life cycle adjustments including consolidation of phases and stages and use of prototyping techniques, and special considerations for testing, validation, and maintenance of knowledge bases. Refer to the *OSWER Project Management Plan Practice Paper* for more information.

3.5 ACTIVITIES

Activities that result in the products listed in Section 3.4 must be completed by the project management and development team. Involving the users and gaining management commitment are two additional activities that must be completed.

3.5.1 Involving the Users

The Initiation Phase requires a great deal of participation of individuals not specifically involved in developing the expert system. If domain experts are not the ultimate users of the system, the actual users must also be represented in describing the problem.

3.5.2 Gaining Management Commitment

One of the goals of the Initiation Phase is to decide whether to commit resources to solving the perceived problem. This will entail generating the necessary management support. Failure to acquire management support now cannot be compensated for in later phases. Therefore, high level management from all involved organizations should participate in system initiation.

CHAPTER 4
CONCEPT PHASE

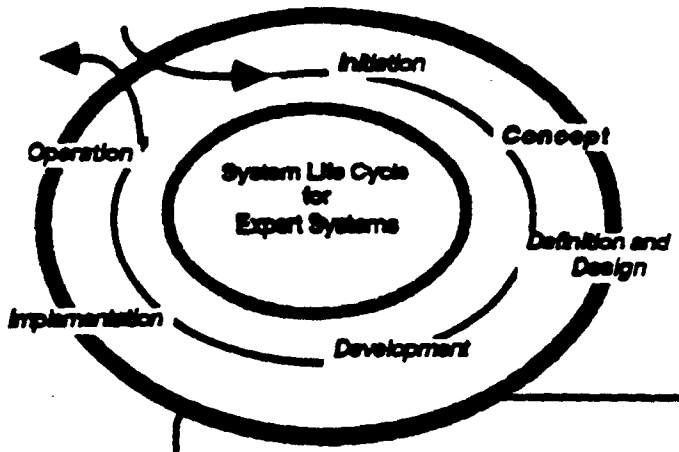
4.0 INTRODUCTION

The Concept Phase is the second of five major phases in the OSWER system life cycle. This phase of the expert system life cycle provides a high-level description of the solution to the problem and describes the functional, knowledge, and data requirements of the task and evaluates alternative solutions. A prototyping exercise may be used to develop the concept, but it is not an end in itself. The Concept Decision Paper is separate from any prototyping activity.

All aspects of the problem solution are considered in the Concept Phase, and many can be addressed by attempting to develop a prototype. Following are the questions which are raised at this time.

- Is an expert system the best solution for the problem?
- What type of feasibility assessment is necessary?
- What knowledge is necessary to solve the problem, and where and how it can be obtained?
- How should the knowledge be managed?
- What features and functionality are expected of the expert system?
- What combination of knowledge representation and control structure best suits the problem?
- How should the project team be assembled?
- What qualifications must the system developers and reviewers meet?
- How can the use of expert systems to solve the problem be justified?

Because several key decisions are made in this phase, it is important to review the objectives of the Concept Phase that are described below.



Objectives

- Determine the feasibility of an expert system solution to the problem
- Identify a cost-effective system solution to the problem

Decisions

- Approach decisions include:
  - Is the problem important and are expert systems a viable solution
  - what is the system life cycle strategy
  - what are the methodologies best suited to the project
  - what is the procurement plan
  - when and from whom will funding be obtained
- Execution decisions include:
  - what are the high level functional and data requirements
  - what are the knowledge management issues
  - what is the knowledge representation and control structure of choice
  - who is to be on the development team and what is their role
  - who are the users
  - can the problem be solved with existing systems
  - what is the overall architecture
  - will the system interface with existing systems
  - what will be the delivery environment
  - how will technical, programmatic, and other risks be addressed
  - what will be the maintenance strategy
- Continuation decisions include:
  - does the information management problem continue to exist
  - does the concept allow the development team to proceed to the next phase
  - are sufficient funding and other resources available

New Products

- System Concept
- Concept Decision Paper
- System Test Document
- Acceptance Test Document
- Knowledge Management Plan
- Data Management Plan
- Requirements Definition

Figure 4.1
Concept Phase
Objectives, Decisions, and Products

4.1 OBJECTIVES

The objectives for the Concept Phase (Fig. 4.1) include a problem definition, requirements, feasibility, and approach. All of the objectives evaluate or describe an approach to solve the information processing problem. The first objective of the Concept Phase is to confirm the existence of the information processing or knowledge-intensive problem.

The second objective is to identify high level requirements for a solution to the problem. These requirements should focus on the nature of the problem and the user's needs, and not directly address expert system issues.

The third objective is to determine the feasibility of an expert system solution to the problem. This requires a study of both the applicability of expert systems to the project and the capabilities of other information technologies. The final objective of the Concept Phase is to identify a feasible, cost-effective expert system approach. In order to meet these objectives, several decisions must be made concerning the approach to, execution of, and continuation from the Concept Phase.

4.2 DECISIONS

Decisions within the Concept Phase (Fig. 4.1) focus on selecting development methodologies, assembling resources, and evaluating the expert system approach to solving the problem.

4.2.1 Approach

There are five decisions to be made in the approach to the Concept Phase. The approach decisions select development methodologies and determine the procurement plan.

4.2.1.1 Approach Evaluation

First, is an expert systems a viable solution? This emphasizes the need to study alternative approaches.

4.2.1.2 Life Cycle Strategy

Second, what is the system life cycle strategy? This decision includes the dependence of the project on rapid prototyping.

4.2.1.3 Development Methodologies

Third, what knowledge acquisition, development, testing, and maintenance methodologies are best-suited for this project? It is important to be aware of the effects of each methodology on the development process, and choose methodologies best suited to the

application. To some degree those choices may emerge from the prototyping approach.

4.2.1.4 Procurement Plan

The fourth decision is determining the procurement plan. How are funds to be obtained and allocated for developing the expert system? Does the project manager need to acquire any hardware or software?

4.2.1.5 Funding Determination

Finally, what portion of the development cycle should be funded at this point and by whom? Is funding available for the entire project?

4.2.2 Execution

There are several decisions to be made in the execution portion of the Concept Phase, most of which focus on the high level requirements of the expert system and resource assembly.

4.2.2.1 Functional Requirements Definition

The first decision during execution is determining the high-level functional requirements of the system. This decision focuses on the nature of the problem and the users' needs rather than on specific expert system issues.

4.2.2.2 Knowledge Management Issues

The second decision involves identifying the knowledge management issues for the project. Are there other expert systems in the same problem solving area that might provide insight? How should the knowledge for this system be acquired, placed into a knowledge base, and maintained?

4.2.2.3 Knowledge Representation and Control Structures

Third, what is the knowledge representation and control structure of choice? If there are no restrictions on the choice of a development environment or expert system shell, which of the knowledge representations is best suited for this particular problem? Given the knowledge representation, which control structure provides the best inferencing? Refer to CHAPTER 5 for explanations of control structures.

4.2.2.4 Data Requirements

Fourth, what -- if any -- are the data requirements and design parameters for this system? Is the expert system supposed to access

information from external data bases or programs? If so, what is the data's format, where does it reside, and how can it be accessed? Some expert systems acquire all data by asking the user for information

4.2.2.5 Project Team Assembly

The fifth decision focuses on assembling the project team. Who is on the development team and what is their role? It is important to know both the qualifications and the availability of potential team members.

4.2.2.6 User Needs Determination

The next decision involves identifying the users, their needs, and their level of sophistication. This is important for properly conducting the feasibility assessment and determining the functional requirements for the expert system.

4.2.2.7 Technology Selection

The seventh decision looks at the need for an information processing system in general. Can the problem be solved with existing systems? If so, then it may be more effective to modify existing systems than to build a new one. If an expert system is chosen, then which shell or language is appropriate?

4.2.2.8 Architecture Planning

The eighth decision entails planning the overall architecture of the expert system, including the structure of the knowledge and the data.

4.2.2.9 Integration Issues

To what degree the expert system will interface with or be integrated into existing information-processing systems.

4.2.2.10 Delivery Environment Determination

What elements of the system will be centralized and what elements will be distributed? This is important in selecting a development environment, determining licensing fees, and so on.

4.2.2.11 Organizational Issues

How will technical, programmatic, and other risks be addressed? Determining the procedures for handling risks also affects the development process and the focus of the expert system.

4.2.2.12 Maintenance Strategy

The final execution decision is selecting the maintenance strategy. Who will maintain the system? How will users' comments be incorporated? How often will the system be updated? Answers arrived at here may be changed later. Refer to CHAPTER 7 for more details.

4.2.3 Continuation

There are three decisions in the Concept Phase that are a continuation of decisions made during the Initiation Phase. The continuation decisions require an objective evaluation of the nature of the problem, the proposed solution, and the resources available to the project.

4.2.3.1 Problem Continuation

First, does the information/knowledge management problem continue to exist? If yes, proceed with the project as it is currently defined. If not, consider refocusing or discontinuing the project.

4.2.3.2 Solution Adequacy

Second, does the expert system concept address the problem sufficiently to permit continuing to the Design and Definition Phase? If it does, then continue to the next phase. If it does not, then look for alternative expert system solutions -- including hybrid expert system/conventional system applications -- or consider other information processing technologies. Prototyping may be used to seek answers to these questions.

4.2.3.3 Project Funding

Finally, are sufficient funding and other resources available for the system life cycle? If resources are available, proceed to the next phase. If not, look for additional resources or postpone the project until they become available.

These decisions are documented in several new products. The products of the Concept Phase are listed below.

4.3 PRODUCTS

There are seven products in the Concept Phase (Fig. 4.1):

- System Concept
- Concept Decision Paper
- System Test Document
- Acceptance Test Document

- Knowledge Management Plan
- Data Management Plan
- Requirements Definition.

4.3.1 System Concept

The System Concept is the key document of the Concept Phase. It describes the results of the functional analyses and both the data and knowledge requirements for the expert system.

4.3.2 Concept Decision Paper

The Concept Decision Paper should explain the benefits of selecting the expert system approach that were determined in the Concept Phase. This document should be clear and comprehensive so that OSWER managers can make an informed decision whether to approve a project.

4.3.3 System Test Document

The System Test Document presents information on the testing to be performed by the development team. It provides a chronology of the system testing process, including strategy, plan, data and knowledge, methods, procedures, results, and recommended actions.

4.3.4 Acceptance Test Document

The Acceptance Test Document presents information on testing to be performed by OSWER program personnel. At the end of the Concept Phase, the Acceptance Test Document contains only the testing strategy.

4.3.5 Knowledge Management Plan

The Knowledge Management Plan describes the approach to acquiring, utilizing, maintaining, and reusing knowledge throughout the project. In the Definition and Design Phase it will include the development team's choice of knowledge acquisition methodologies, knowledge representation, control structure, and maintenance strategy.

4.3.6 Data Management Plan

The Data Management Plan reflects the project's data management approach. This document supplements the Knowledge Management Plan by describing access to external data in databases, application programs, and historical files.

4.3.7 Requirements Definition

Information from the experts and users are compiled to form the guidelines used in the Definition and Design Phase of the expert system development life cycle. Information to be included in the Requirements Definition specific to expert systems are:

- Target level and focus of output
- [Explanation facilities]
- User interface
- External interfaces
- Target performance (speed and accuracy) of the system.

4.4 SUCCESS FACTORS

Several factors that affect an expert system's success should be considered in the Concept Phase. They fall under the general topic areas of organizational and resource issues, the target users of the system, functional requirements, and knowledge representation and control structure. Several success factors are identified and general advice is given on applying them.

The first major success factor is specifically identifying and effectively implementing the products of the Concept Phase. Proper utilization of the Concept Phase leads to advantages such as assembling an efficient team, good Requirements Definition and resource estimation, and clear management direction.

Another high-level success factor in the Concept Phase involves implementing expert systems around well-conceived ideas. These can be developed through prototyping, but this is not a substitute for the concept. Expert systems should be implemented to solve problems that are not effectively handled using conventional technologies. This success factor can be ensured by carefully studying the problem and looking for conventional alternatives.

Success factors relating to specific areas within the Concept Phase are described below.

4.4.1 Organizational Issues

There are several organizational issues that affect the outcome of an expert system project. Organizational issues include scheduling flexibility, management commitment, available hardware and software availability, and implications assessment.

4.4.1.1 Scheduling Flexibility

The first organizational issue is that the concept of an expert system is not critical to the solution. This implies that there is

freedom for the project to change directions and use a solution other than an expert system. System flexibility is important for expert system projects because they frequently need major modifications as ideas transform and new directions are discovered, either through prototyping or conventional analyses.

4.4.1.2 Management Commitment

Management commitment to an expert system project is often cited as one of the most common reasons for project success. Management commitment comes in many forms: resources -- including dollars, people, and equipment; continued involvement and supervision; and support in times of conflicting objectives. Management needs to be aware of the benefits, limitations, and differences between expert systems and conventional information processing tasks at the initiation of the project (see CHAPTER 3). A constant flow of information and updates is required to keep management involved, interested, and committed to the project.

4.4.1.3 Impact Assessment

Another organizational issue is that the ramifications of the expert system -- on people, the task, or the organization itself -- are thought out in advance. Take time to review the implications of changing current processes caused by adding an expert system.

4.4.2 Resource Issues

Major resource issues stem from the proper evaluation and resource estimation of expert system projects. While this is a difficult task, proper evaluation of the project goes a long way to ensure that the resources are best allocated to an expert system, and would not generate higher payoffs on other projects or other technologies.

Resource estimation is a critical factor influencing the success of expert system projects. It is important that the developer accurately estimate the time, staff, and financial resources required to complete the project. Another resource estimation factor is allowing for experimentation or exploration through prototyping, both often necessary in the development of an expert system. In order to benefit from these factors, the project team should evaluate other expert system projects and set aside extra funds for necessary exploratory work.

4.4.3 Measures of Success

Measures of success for the expert system project are clearly stated and agreed upon. Measures of success can be quantitative -- increased productivity, or time savings -- or they can be qualitative -- such as improved morale. These measures need to be identified and defined prior to

the start of the project so that they are used to guide and evaluate the expert system project.

4.4.4 Target User Level of the System

Success factors associated with the target user level of the system involve an understanding of the users' needs and a specification of the content and level of complexity of expert system's outputs. Some effort should be put into determining how the expert system's recommendations will be used.

4.4.4.1 User Identification

The first factor is identifying the intended users of the expert system. This leads to a clear idea of both the focus of the output and its level of sophistication. Users at an entry level position will require a different focus -- one that pertains directly to their task -- as well as a specific degree of sophistication. The output should be very specific and in terms that they can understand. Advanced users, on the other hand, are often better served by succinct answers that they can use as guidance.

Identifying the intended users is also essential when they vary in their degree of computer literacy. Special help features may be necessary depending on the level of computer experience of the target users.

Another success factor in determining the target output of the system is the degree of training that is necessary. If the expert system is targeted entirely toward training, then it should focus on providing as much information to the user as possible. Training systems often try to diagnose what problems a user is having with a concept, and then set up exercises to correct the problem. Expert systems that are intended to have only incidental training benefits focus on solving the problem with a minimum amount of overhead and their output tends to be more succinct.

4.4.4.2 Content of System Outputs

A second facet of this issue is understanding how the output of the system is to be used. Once the users are identified and the target level of the output is set, how are the users to treat the expert system's recommendations? Are the recommendations to play the role of a checklist, an assistant, or an expert? It is critical that the users of an expert system understand the degree to which they can rely upon the output of the expert system and the level of their own responsibility.

4.4.4.3 Level of System Outputs

The final success factor in the target output involves specifying the underlying knowledge that will be incorporated into the system. Given the application, what information is necessary to make an informed decision to accurately solve the problem? It is important to gather this information from the expert before beginning the next phase.

4.4.5 Functional Requirements

It is important that functional requirements are carefully defined. It is quite difficult to fulfill moving specifications under any circumstances and especially given time and resource constraints. The requirements should be written, agreed upon, and modified only when the time and resources are also changed commensurately.

Justifications for the expert system's recommendations need to be clear and specific. Explanation facilities -- such as the why and how queries often found in expert systems -- often consist of replaying the logic used by the system to arrive at a conclusion. While this is sufficient for some applications, others require more in-depth justification including causal relationships and assumptions made.

4.4.6 Knowledge Representation and Control Structure

When an expert system shell is already selected before the Concept Phase is completed, the choice of a knowledge representation and a control structure is somewhat limited. This poses a challenge to the developers when the nature of the problem does not fit well with specific knowledge representation scheme supported. The feasibility study should show whether or not alternate development environments would provide a cost-effective means to successfully avoid this problem.

The proper knowledge representation and control structures expose the natural constraints of a problem and allow the expert system to be built efficiently and easily. At times, such as when the knowledge representation is predetermined by the available expert system shell, there may be no alternative but to proceed. When there is a choice, the nature of the problem should be used to select a suitable knowledge representation and control structure.

4.5 ACTIVITIES

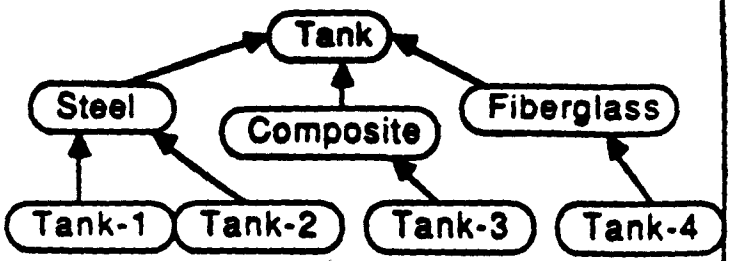
There are several introductory activities to be undertaken during the Concept Phase. These involve three types of participants in an expert system project - the users, the expert, and the project management. These activities will lay the groundwork for the major activities of this phase.

Knowledge Representations

RULE

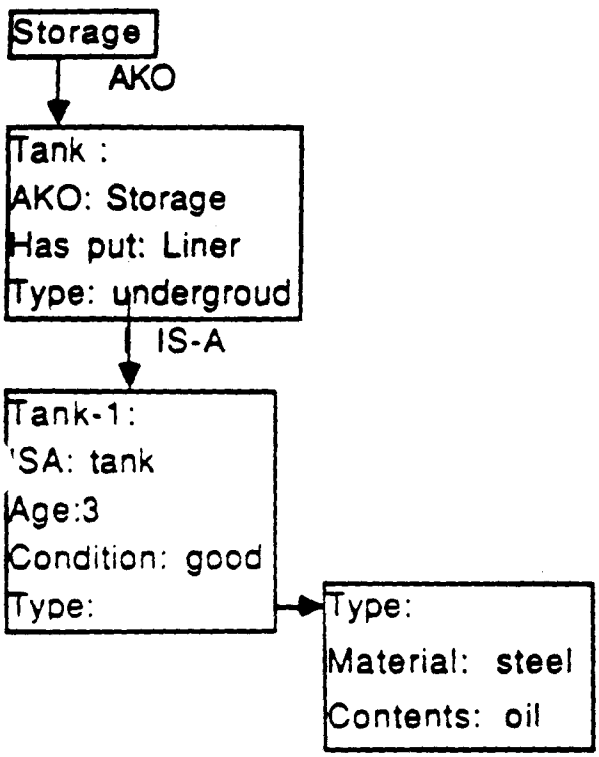
IF the UST has not been upgraded
AND age is greater than 10 years
THEN upgrade must be performed

OBJECTS



Messages: create, destroy, upgrade

FRAME



OAV Triplets

object	attribute	value
tank	type	steel
tank	type	fiberglass
tank	age	10

Hybrid-Rule-OAV

IF NOT Tank:condition:upgrade
AND Age > 10 (parameter)
AND Tank:type:steel
THEN upgrade UST

Figure 4.2

4.5.1 Identifying the Users

Identifying the users of the application seems apparent, but in fact can be quite difficult. [End user] identification is critical to proper design and development because several expert system features are determined by the intended end users, including:

- The delivery environment and geographic distribution,
- Target level of the expert system's recommendations,
- Type and sophistication of the user interface,
- Level of explanation in the justification mechanism, and
- The focus of the knowledge base itself.

The ultimate users will work with the expert system on a regular basis. Other people who will be indirectly involved with the use of the expert system -- such as managers, support technicians, and maintenance staff -- are not considered users. A precise definition of the users leads to a more focused and productive expert system.

After all of the people who will be directly involved in the operation of the expert system are identified, a profile of the group should be developed. This profile contains information on the specific task that the users need the expert system to perform, the level of job training and experience of the users, and their range of computer literacy. The conceptual model of the expert system can then use this user profile to specify features that focus on the specific needs of the intended users.

4.5.2 Selecting the Expert

Selection of the expert may seem like an obvious task, but should be given adequate consideration. Several factors should be considered when selecting an expert, or determining if an appropriate expert exists. This generally leads to an expert who is cooperative and easy to contact and schedule appointments for [knowledge engineering] sessions. The knowledge engineer should also elicit support and commitment from the expert's superior.

The expert needs to have the requisite qualities to facilitate knowledge engineering. The expert should be methodical, consistent, and articulate in dealing with the knowledge engineer. With this type of expert, it is easy to get the expert more interested in the technology.

The expert must not be intimidated by the thought of an automated system doing their job. Explain to the expert that the system is not meant to be a replacement, but to aid the expert in making more informed decisions or to help people in the field where the expert might not be available. The expert should understand what is required. It is the responsibility of the knowledge engineer to properly inform the expert what is expected.

4.5.3 Communicating with Management

Management support is crucial to the success of any information-processing project and especially for expert systems. Many people are uncertain of the capabilities and practicality of expert systems. The way to obtain and maintain management support is to provide good channels of communication in the Concept Phase and throughout the life cycle.

In the Initiation Phase, management needs to be informed of the capabilities of expert systems: their benefits, limitations, and feasibility. Expert systems should be presented to management in terms of improving productivity in current work or in the capability to accomplish tasks that were previously not feasible. Introduction to expert systems is often best accomplished via written material followed by a briefing. This allows the managers to learn about expert systems at their own pace, with a minimum of time commitment. Next, management needs to understand the specific expert system application area. This will improve the quality of decisions made throughout the life cycle of the expert system.

Throughout the development process management needs to be kept informed on the status and needs of the project. It is best to use written memos and reports to keep an audit trail of the development process (see Section 7.6.9). Another useful form of communication is the demonstration. An expert system application already in use can be used to demonstrate the general features of an expert system. Demonstrations of working prototypes should be given for management throughout the development cycle to illustrate the progress and the direction of the project.

4.5.4 Management Commitment and Understanding of Prototyping

Management commitment is the key to the success of an expert system project. There must be a perceived value in excess of perceived cost and risk in the project to retain management commitment. Managers must also be aware of the different techniques used in developing expert systems. Techniques used in expert system development that may be unfamiliar to managers include knowledge acquisition, prototyping, and knowledge base verification.

4.5.4.1 Initial Management Commitment

The need for management commitment is obvious, but is often overlooked. Initially, management commitment involves approval and initiation of the expert system project. Management commits the resources to begin the project, including funding for hardware, software, and human resources.

4.5.4.2 Continued Management Commitment

Continued management support as development proceeds is sometimes more tenuous, and in some ways more important. Once the project has begun, there is a tendency to reallocate people on the expert system project when conflicts arise. An important form of management commitment is supporting the expert system project even when budget cuts or other resource constraints are imposed. Once the project has started management needs to be objective when comparing the expert system project to other information-processing projects, and not be adversely influenced by the mystique of expert systems.

4.5.4.3 Prototyping Issues

One of the major benefits of prototyping techniques is their inherent suitability to expert systems. Expert system prototypes can be developed, evaluated, and modified relatively quickly and inexpensively. This allows additional opportunities to be explored with minimum risk. Managers need to be aware of the ramifications of prototyping (see Section 1.7) and be prepared to utilize prototypes to their fullest.

4.5.4.4 Life Cycle Issues

Because of the significant differences between the development processes of traditional information processing projects and expert systems -- such as the disposable prototype, management needs to be alerted to expert system life cycle issues. Major issues include knowledge management, knowledge acquisition, prototyping, and knowledge base validation. Proper understanding of these issues (see CHAPTER 1) will allow managers to reap the full benefits of expert systems.

4.6 KNOWLEDGE MANAGEMENT

"In the knowledge lies the power" is a quote by Randall Davis of the Massachusetts Institute of Technology, who has helped develop several famous expert systems such as MYCIN. In this statement he emphasizes the importance of the knowledge placed in an expert system. Knowledge is more than information, just as information is more than data. Knowledge results from the capability to use information. A planned methodology is necessary to properly gather, use, and maintain knowledge for expert systems. Knowledge management comprises six parts, described below.

4.6.1 Knowledge Acquisition

In the knowledge acquisition process, we identify, locate, and gather the information and mental processes used to solve problems. The first step in knowledge acquisition is to determine what knowledge is necessary to

solve the problem, including any background information. The required knowledge acquisition information includes the:

- Knowledge necessary to solve the problem;
- Location of the knowledge -- including experts, texts, historical data, and test data;
- Selected knowledge acquisition methodology; and
- Plans for storing and manipulating the knowledge.

4.6.2 Knowledge Representation

Knowledge representation is important for four reasons. First, it simplifies the task of accessing the acquired knowledge in the system. Second, a good knowledge representation scheme exposes the natural constraints of the problem, which makes it easier to solve. Third, the knowledge representation is the media in which the knowledge is stored, updated, and modified. Thus a good knowledge representation will expedite maintenance. Finally, selecting the proper knowledge representation will facilitate knowledge transfer by making the knowledge transparent and easy to manipulate. The information needed for the Knowledge Management Plan from this part is the preliminary choice of knowledge representations for the expert system.

4.6.3 Knowledge Base Creation

Knowledge base creation consists of taking the acquired knowledge, placing it in a knowledge dictionary, transforming it into the selected knowledge representation, and placing the acquired knowledge into the knowledge base.

4.6.3.1 Knowledge Dictionary

The knowledge dictionary for expert systems is similar to the data dictionary for databases. The purpose for the knowledge dictionary is two-fold. First, it documents the structure of the knowledge base contents. This facilitates both testing and maintenance. Second, the knowledge dictionary enforces consistency in naming and transforming knowledge. This is especially important when several programmers are working simultaneously on the knowledge base.

4.6.3.2 Knowledge Transformation

The transformation process is not always straight-forward because of the inherent restrictions and constraints of any knowledge representation. In addition, there are concessions that have to be made for operational purposes. These concessions generally consist of additions to the knowledge base that are solely for the purpose of controlling the flow of operation, calling external data sources,

or updating user interfaces. These additions are necessary for the operation of the expert system, but can cause confusion in later attempts to utilize the information in the knowledge base. Assumptions are also placed into the knowledge base based the way the control structure will operate. These assumptions add to the problems of maintaining or reusing the knowledge, and should be minimized.

4.6.3.3 Knowledge Modularization

It is particularly important to properly modularize the knowledge base. Knowledge about a problem domain can usually be decomposed into smaller components. For example, in a diagnostic expert system for a piece of equipment, the problem might decompose into diagnosing the electrical system, the mechanical system, and the structural portion of the equipment. Each piece of the problem can be placed in a separate module within the knowledge base.

Modules within knowledge bases serve several purposes. First, they allow common information to be stored in one centralized area. Second, they allow the developer to think about the sub-problems independently and thus simplify the development process. Third, modules facilitate the verification process by isolating errors. Fourth, maintenance is simplified because updates and corrections are made to independent and easily identified parts of the knowledge base. Finally, software reuse is promoted by separating knowledge.

4.6.4 Knowledge Base Validation

Validation of the knowledge base is necessary to ensure proper performance of the expert system. A knowledge base validation plan helps developers perform a comprehensive review of the expert system and its capabilities. Information on when the validation is to occur, the type of verification to be performed, and where the test data is to come from should be included in the plan.

It is important to realize that value can be derived from any expert system project, even those that are not implemented. The knowledge acquired in building an expert system is at times more valuable than the expert system itself. This is due to the fact that the problem-solving process is now documented, and can be reviewed and stored.

4.6.5 Knowledge Base Maintenance

The need for maintenance arises from changing or evolving user requirements, the need to correct errors, revisions in regulations or procedures, and advances in the state-of-the-art. The maintenance plan must recognize these sources of change and be prepared to react accordingly. Once the changes are made, there should be a knowledge maintenance plan and

configuration management plan for testing and validating the knowledge base to ensure its continued usability.

4.6.6 Knowledge Base Reusability

The ability to reuse knowledge bases saves resources. Knowledge management should stress the reuse of knowledge to the extent that it is a viable alternative. A major portion of the creation of a comprehensive knowledge base is the inclusion of background information and underpinning knowledge. Once this information is stored in the knowledge base, the knowledge to solve the specific problem is easy to enter. Because of the relatively high ratio of background knowledge to application-specific knowledge, there is an evolving process of reusing knowledge for expert systems performing in the same problem area.

Knowledge reusability is still a research issue at this time. Several large projects are underway to develop general-purpose knowledge bases, but practical applications are limited in number. There are some aspects of knowledge reusability that might be helpful to OSWER project managers:

- Identifying other expert system projects within the same problem-solving area,
- Obtaining these systems and their documentation,
- Conducting an extensive literature search,
- Building a well-structured, modular knowledge base,
- Maintaining the knowledge base, and
- Using simple versus compound knowledge representations.

4.6.6.1 Other Potential Expert Systems

The developer should be aware of other potential expert systems within the same problem-solving area. If others have been developed, try to obtain a copy of the knowledge base or the knowledge acquisition sessions. If other expert systems are planned, the knowledge engineer needs to obtain as much general knowledge as possible in the sessions, document assumptions and paths not taken, and store the results of the knowledge acquisition in an easily accessible manner.

4.6.6.2 Modular Development

The second step in practical knowledge reusability is to build a well-structured knowledge base. This implies making the knowledge base modular and as complete as possible. Assumptions should be

recorded, and the entire knowledge base well documented both on-line and in hard copy. Inclusion of procedural pieces and operational code should be kept to a minimum. Finally, the knowledge base should strive for clarity rather than efficiency or elegance.

4.6.6.3 Maintenance Procedures

The final step of knowledge reusability is in the maintenance of the knowledge base. Maintenance should adhere to the same principles as the development, or its reusability will decrease over time.

4.7 FEASIBILITY ASSESSMENT

A feasibility assessment is essential to understand the scope of an expert system project and the costs and risks associated with it. This feasibility study parallels that of the *LCM Guidance*. A feasibility assessment should be conducted by the knowledge engineer(s) in conjunction with the expert(s) and the intended users. The issues involved in the feasibility assessment process are technical in nature. The objectives of this process are to determine the suitability and to define the functional requirements of the proposed expert system.

4.7.1 Determining the Viability of an Expert System Solution

The first step in determining the feasibility of an expert system project is to determine the need for an expert system. After the problem itself is verified as being important, the approach to solving it is studied. If other solutions exist, it is important to consider whether expert systems would add sufficient benefits to justify their use. Another consideration in this step is the possibility of combining expert systems with other computing techniques. Problems involving forecasting, optimization, or voluminous information management might be best solved using a hybrid approach.

4.7.2 Technical Issues

Technical issues to consider in the feasibility assessment include the nature of the problem, characteristics of knowledge management, availability of expertise, software interfaces and integration, and verification and validation.

4.8 KNOWLEDGE REPRESENTATION

Knowledge representation is recognized as a crucial part of any artificial intelligence project. Choosing the correct knowledge representation facilitates the development and maintenance of an expert system. Figure 4.2 provides a graphic representation of the following aspects of knowledge representation.

4.8.1 Rules

Rules are the most common form of knowledge representation for expert systems, following a simple if-then format. Rules are used to represent the rules-of-thumb or [heuristic] used in solving problems. People find it easy to describe complex processes in simple steps using if-then rules. Rules are good for capturing procedural knowledge. Because they capture only one small piece of the problem per rule, rules are not applicable to all problems.

4.8.2 Frames

Frames resemble database records in that they store chunks of information together in fields. They differ from records, however, by adding class-subclass structure, including more information to individual fields, and by incorporating procedural inferencing mechanisms into the information.

Frames are used to store knowledge that has an important structural component (e.g., hierarchies of chemical information.) Frames are often used to store case histories, because they store relevant information in the same place which facilitates retrieval.

4.8.3 Objects

Objects are similar to frames, except that they normally rely on message-passing for communications and are more independent. Objects tend to be used in expert systems in a limited form, as scaled down versions of frames.

4.8.4 Object-Attribute-Value Triplets

Object-attribute-value (OAV) triplets and parameters are a simple form of knowledge representation used for storing rudimentary information. They are generally used in combination with other knowledge representations such as rules.

OAV triplets store information in three parts. The object portion contains the name of the primary concept that is being represented by the triplet (e.g., an apple). The attribute portion contains information on what particular aspect of the object we are referring to (e.g., the color). The value stores the actual information describing that particular attribute of the object (e.g., red).

4.8.5 Parameters

Parameters are similar to the variables of various programming languages in that they are typed and contain values. Different types of parameters include numbers, strings, sets, lists, and records. The difference is that parameters as a knowledge representation also contain information about their use that is utilized by the expert system such as an explanation

of what the parameter represents, questions to ask the user to find the value for a parameter, and information on where the parameter is used. Parameters store basic knowledge that is neither procedural nor structured.

4.8.6 Hybrids

Hybrid knowledge representations are becoming more prevalent because they make up for the deficiencies of individual knowledge representations. Hybrid knowledge representations typically consist of a combination of rules and frames, or rules and objects. This combines a structural storage representation frames or objects with action-oriented representation rules to capture complex knowledge more easily across a broad range of problem domains.

4.9 CONTROL STRUCTURES

More than one control structure may be applied to most knowledge representations. The selection of a control structure depends on the nature of the problem, the knowledge representation chosen, and the desired performance of the system.

4.9.1 Forward Chaining

[Forward chaining] applies to the rule knowledge representation. It takes data and information, applies the appropriate rules from the knowledge base, and presents recommendations. Because of this, forward chaining is also known as data-driven and event-driven processing. Forward chaining is best for problems that have a specific set of inputs and a large number of possible outcomes including design and configuration applications.

4.9.2 Backward Chaining

Backward chaining also applies to rules, and works backward from the conclusions until it finds a combination of rules and data that support a given solution. Backward chaining works best on problems that have a limited number of possible outcomes and several inputs including diagnostic and classification applications. Backward chaining results in intelligent questioning, because only information pertaining to the current hypothesis is requested.

4.9.3 Inferencing

Inferencing is a general term that refers to several other control structures, including pattern matching, [backtracking], constraint propagation, and search techniques. (These terms are defined in the glossary in APPENDIX B.) Variations on these techniques are used in expert systems that use knowledge representations other than rules, such as frames and objects.

4.9.4 Hybrids

Hybrid control structures attempt to reap the benefits and overcome the limitations of one or more individual techniques. Common hybrid control techniques combine forward and backward chaining, or chaining and pattern matching. The hybrids are typically applied to large problems and applications using hybrid knowledge representations.

4.10 THE SYSTEM CONCEPT

The conceptual model contains comprehensive information concerning the details necessary to properly design the expert system. Information contained in the conceptual model includes;

- Specific problem area and solution type,
- Knowledge representation,
- Control structure,
- External data sources and interfaces,
- User interface,
- Target output level, and
- Justification mechanism.

The conceptual model should also be the basis for making a preliminary assessment of the expert system shell or development environment subject to the Definition and Design Phase in Chapter 5.

4.11 PROOF-OF-CONCEPT PROTOTYPE

Prototyping is an iterative process of design and redesign during development. Each prototype builds on the successful completion of the previous prototype. Prototyping involves several stages, the first of which begins in the Concept Phase. The Proof-of-Concept Prototype is initiated during this phase.

The Proof-of-Concept Prototype is a very small working model of the expert system developed to assess preliminary feasibility of the problem domain. It is developed by following a narrow line of reasoning for a specific topic to its conclusion. If using rules as a knowledge representation, the number of rules should range from 5 to 25. This prototype shows inherent strengths and weaknesses of further development. It will answer the question of whether or not another technology or approach is feasible. It also allows the knowledge engineer to assess the framework, scope, interfaces, and issues borne out during knowledge acquisition and the design process.

4.12 ASSIGNING A PROJECT TEAM

The project team consists of everyone directly involved in the development of the expert system. The aspects of project staffing addressed here are the size and composition of the project team. The size and

composition of the project team will be directly related to the nature of the problem and the conceptual model developed earlier in this phase. The number of people in project management will remain approximately the same for expert system development.

4.12.1 Expert Involvement

Experts can vary in number from one to 10 or 50. Fewer experts are needed on small projects that have adequate development time. In the case of specialized projects, only a few experts may be available. Multiple experts are used on large projects, and particularly those combining expertise in several related areas. Multiple experts might also be used in parallel on projects that have limited development time. It is important to not use all of the available experts in developing an expert system. Some should remain external to the project until it is time to verify the expert system's knowledge base.

4.12.2 Knowledge Engineer Involvement

Knowledge engineers are key members of the expert system development team. There should be one senior knowledge engineer and one or more junior knowledge engineers depending on the chosen knowledge acquisition methodology and on the number of experts. Some knowledge acquisition methodologies such as the two-on-one interviewing technique require multiple knowledge engineers. Except for extremely small expert system projects, there should be multiple knowledge engineers to increase accuracy and completeness of the knowledge acquisition and transformation process. In projects where multiple experts are involved at the same time, a guideline is to have two knowledge engineers per expert.

4.12.3 Programmer Involvement

The number of programmers on the development team depends on the role of the knowledge engineer and the amount of conventional programming that needs to be performed. If the knowledge engineer is capable of and available for programming the expert system, then programmers per se may not be necessary. Programmers are important for integrating the expert system into existing operations and for adding external functions to the expert system especially hybrid expert/conventional systems. Estimating the number of programmers is otherwise the same as with conventional systems.

4.12.4 Reviewer Involvement

The number of reviewers is generally higher for expert systems projects due to the need for external experts for knowledge base verification.

CHAPTER 5
DEFINITION AND DESIGN PHASE

5.0 INTRODUCTION

After an appropriate problem domain has been selected, the Definition and Design Phase begins. This phase is critical because it lays the framework for the development activity to follow.

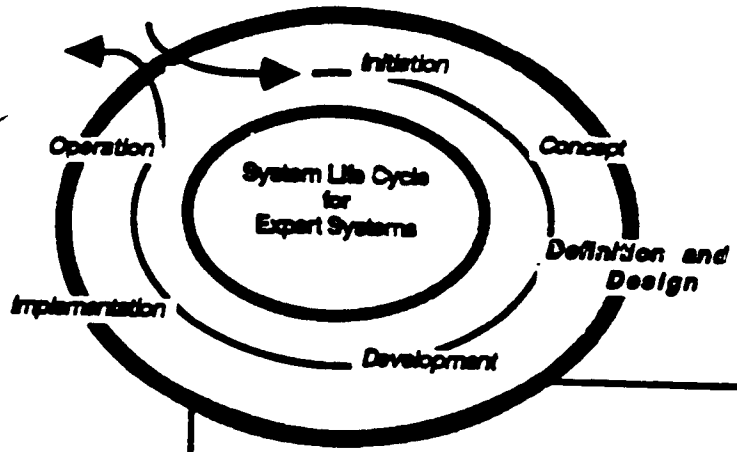
5.1 OBJECTIVES

The following objectives of the Definition and Design Phase (Fig. 5.1) must be met in order to assure a smooth Development Stage.

- Problem definition - the definition of the problem domain has to be refined and reaffirmed.
- Development environment selection - the success of the Development Stage hinges on careful selection of the development environment. The goal is to select an environment that is a good match for the problem domain. Refer to Section 5.5 for more details.
- Delivery environment selection - the consideration of the delivery environment is also an important objective of this phase. The requirements of the end users should be incorporated in the early stages of system design.
- Evaluation of expert system shells applicable to the problem domain.
- System design - ultimately, an overall design of the system should be achieved.

5.2 DECISIONS

Several decisions must be made in the Definition and Design Phase (Fig. 5.1). The main decision involves the choice of an appropriate development environment. The importance of a good selection at this stage cannot be stressed enough. Refer to Section 5.5 for guidelines in the selection of a proper development environment. Also, the targeted delivery environment must be carefully selected to provide a smooth transition from development to implementation. In addition, teams should be assembled to design and develop the system.



Objectives

- Refine the problem definition
- Select a development environment that is compatible with the problem domain
- Select the delivery environment
- Achieve a satisfactory overall system design

Decisions

- Decisions to be addressed include:
  - selection of an appropriate development environment
  - selection of a targeted delivery environment
  - identification and assembly of design team members
  - identification and assembly of development team members

New Products

- One-page Design of the System
- Detailed Design of the System
- Development Plan
- Test and Validation Plan
- Support Plan
- Development Environment Evaluation

Figure 5.1

Definition and Design Phase
Objectives, Decisions, and Products

5.3 PRODUCTS

There are six products developed during the Definition and Design Phase. These include design documents, management plans, and evaluation.

5.3.1 One Page Design Document

The One-Page Design Document is a brief description of the expert system. It gives the expert systems scope, purpose, and a preliminary assessment of the knowledge representation and control strategy which applicable to the problem domain. This document is described fully in 5.7.1.

5.3.2 Detailed Design Document

The Detailed Design Document is an expansion of the One Page Design Document. Further refinement is made of the items in the one page design document. Refer to Section 5.7.2 for a further discussion of this document.

5.3.3 Development Plan

This plan is derived from the detailed design document. It outlines the resources needed for the Development Stage. Also, schedules are firmed up in this plan.

5.3.4 Test and Validation Plan

This plan is developed to specify how testing and validation should be handled. It will contain a methodology for designing tests for the expert systems. It will also list resources available during testing and specify when testing and validation will take place. Key people in the validation process are named in the plan.

5.3.5 Support Plan

This plan contains information on what support is available. All resources and the time they are needed are specified in this plan. Refer to Section 2.5 for information on what resources are necessary in the Definition and Design Stage.

5.3.6 Development Environment Evaluation

This document outlines potential development environments. It lists strengths and weaknesses of each. The factors that need to be addressed in this document are primarily from the detailed design document. Refer to Section 5.5 for procedures for selecting a development environment.

5.4 SUCCESS FACTORS

During the Definition and Design Phase, both the development and delivery environments must be considered as integral components of the system design. Some problems fit neatly into commercial shells, while others do not. The various development environments offer a variety of mechanisms for knowledge representation, control structure, and conflict resolution. These must all be evaluated with care to avoid problems later during the Development Stage. Common expert system design success factors are listed below.

5.4.1 Validate the Development Environment

The desired development environment features selected in Section 4.11 are validated during this phase. Various software, hardware, and technical issues should be addressed in the selection of a specific product.

5.4.1.1 Hardware and Software Availability

The first consideration in selecting a development environment is determining the hardware and software currently available to the user base. This impacts, and may highly constrain, the choice of both development and delivery environments.

5.4.1.2 Software Issues

Explore the wide variety shells first to try to find a match with the problem domain. Shells can provide a quick solution to many problems. Refer to Appendix A for expert system shell evaluation factors. Do not attempt to use complex languages without proper training, appreciation for the difficulty of the task, sufficient time for development, or the availability of experienced programmers.

If a shell has been selected as a development environment, be sure that the tool is affordable, easy to install, and easy to use. An overly expensive tool dilutes the overall benefits derived. Choose a tool that can accommodate the problem. Furthermore, the tool must be able to handle the necessary data. A superior shell can be modified to provide additional features.

5.4.1.3 Hardware and Technical Issues

Special development hardware may be required to run the tool. The additional cost must be considered in the selection process. The development environment should be able to scale up well. Integration with desired data sources, programs, and output interfaces should be readily accommodated. Establish reasonable goals for satisfactory performance in terms of speed and memory use. Migration to a suitable delivery vehicle should be easy and inexpensive.

5.4.2 Verification and Validation

Verification and validation methodologies must be determined at this stage. Verification techniques are the methods used to determine that the expert system has been built correctly. Verification techniques for the software, knowledge base, and interfaces should be derived from the expert system's functional requirements. Steps taken and methods used to verify the expert system need to be mapped from the components up through interfaces and software module interactions. The operational points of the expert system that need to be validated -- such as scope, effectiveness, and compatibility -- are identified at this point.

Validation techniques are the methods used to determine that the expert system conforms to the functional requirements and can be used as intended. The validation techniques should also be identified in the Definition and Design Phase. Issues such as the need for external experts, and types and location of test cases should be thought out.

5.4.3 Delivery Environment

When establishing a design for the delivery environment, there are several things to be aware of. These include:

- The environment's flexibility, availability, and consistency with equipment currently being used,
- Licensing fees required for distribution of the system. Be sure ask the vendor about licensing fees when selecting a shell.
- Graphics requirements for the Production System.
- Problems caused by excessive data migration from the mainframe to a PC. If it appears that there will be excessive data migration, perhaps it is a case when a mainframe implementation should be used.
- The migration from the development environment to the targeted delivery environment may be technically impossible. Careful selection of development and delivery environment will avoid this problem.

5.4.4 Rationale and Justification Features

Several decisions should be made during this phase regarding the necessary rationale and justification features of the development environment. These include:

- Specifying the extent to which help is available in the software and in which areas,
- Clarifying what constitutes an effective rationale for posing questions and for issuing recommendations, and
- Determining a consistent method for ranking recommendations.

5.4.5 User Interface Issues

The end user interface is the most important portion of the expert system. User involvement in selecting the desired features should be included as early as possible to ensure the success of the system. Some issues to address include:

- Incorporating interesting, user-friendly screens and system features,
- Providing convenient data input for the end user without the advanced editing functions needed only in development,
- Ensuring adequate system response time, especially for calculation-intensive applications, and
- Creating query interface capabilities that support flexibility and sophistication.

5.5 SELECTING A DEVELOPMENT ENVIRONMENT

In the Concept Phase, a preliminary assessment of the development environment was made. During the Definition and Design Phase, the actual selection of the development environment is made. Several factors are involved in selecting a development environment, including the:

- Choice of delivery environment (hardware, software, and degree of integration with existing systems),
- Type of problem including whether or not it is a hybrid expert system/conventional system problem,
- Suitable knowledge representations and control structures, and
- Necessary interfaces.

5.5.1 Features

Some of the features of a development environment are outlined to give a better understanding of how to select one.

5.5.1.1 Declarative and Procedural Elements

Knowledge bases can be composed of declarative and/or procedural elements. A declaration is a statement of fact or an assertion that some data item has a given value. An example of a declarative statement is: "The average annual temperature at weather station X is 66 degrees." In contrast, facts may also be derived by procedural methods. The above fact could have been represented by a procedure to compute the average annual temperature from raw data. A knowledge base consisting solely of declarative statements is very explicit. Such a knowledge representation is readily updated by users with minimal training. However, procedural additions can greatly increase the power and flexibility of the knowledge base. Procedures handle abstract data and can allow for changes in information without altering the knowledge base.

An effective tool incorporates both declarative and procedural knowledge representations. Development environments that offer frames often support both types of representations. Frames are groups of slots that can contain declarative statements of facts and/or procedural attachments to determine facts. (See Section 4.8.2. for more information on frames).

5.5.1.2 Grouping and Modularity

Grouping and modularity are two related techniques that allow for faster development and greater extendibility of knowledge bases. If an expert system application is very complex, the problem domain can be separated into modular knowledge bases and the resulting segments grouped into specialized areas. In this manner, each knowledge base or group of knowledge bases can focus on a narrow aspect of the overall problem. To expand the system, the developer can alter a particular subset of the problem domain, while maintaining the integrity of the other knowledge bases.

Validation of an expert system is a much simpler task if each component knowledge base is a manageable size and can be tested independently. The chosen development environment should include the ability to separate a large problem domain into smaller distinct knowledge bases. Furthermore, it is necessary to have some means of communication between the segments, such as parameter passing or a [blackboarding] mechanism.

5.5.1.3 Documentation Needs

Clear documentation of the knowledge base is extremely important during the development of an expert system. Some tools represent the knowledge base in a very readable, English-like format that is almost self documenting. The developer must be able to review the knowledge quickly and easily at all stages of development. Also, concise documentation is necessary for validation and maintenance, especially if these tasks are to be performed by someone other than the developer.

5.5.1.4 User Issues

Another factor involved in the selection of a development environment is its ease of use. This is the result of the user interface, and is determined by the type of interface provided by the development environment and the effort of the developers.

5.5.1.5 Developer Issues

There is often a trade-off between the sophistication or power of an [expert system development environment] and its ease of use. In other words, expert system computing environments that are best-suited for difficult problems are often more difficult for the developer to use. It is important to take into consideration the complexity of the problem and the skill of the developers when selecting a development environment. Trade-offs between development and run-time environments should be made explicit.

5.5.1.6 Knowledge Management Issues

The developer needs to be able to describe the knowledge represented in the system in order to interact with the expert. A careful consideration of what has been included in the knowledge base is essential before any expansion takes place. The current knowledge should be outlined and discussed to expose weaknesses or gaps in reasoning. The knowledge gaps can then be addressed in future development. A tool that cleanly breaks out the knowledge into a hierarchy or decision tree will be of great help in this regard. Also, some tools incorporate data element dictionaries.

5.5.1.7 Rationale

Most development tools allow the user to ask the system for its rationale for asking a particular question during a consultation. The user need only enter "Why?" in response to a system prompt. The system will generally show its current line of reasoning and explain why it needs the information in question. The user may choose to answer the original prompt or perhaps respond with "Unknown" if no

information is available. The ability of a system to allow a series of "Why?" inquires to trace the line of reasoning has great value.

5.5.1.8 Justification

A common feature of development tools is the ability to explain the justification for the recommendations of a consultation. The user can ask "How?" after a consultation to get a detailed trace of the line of reasoning used to arrive at the given recommendation. This trace can be in both text and graphic format, and often allows the user to change one or more responses to experiment with a "What if?" scenario.

5.5.1.9 On-line Help

Many expert system shells offer on-line help both during development and during a consultation. For example, the user can ask the system for syntactic information while developing the knowledge base. This feature is especially useful for the novice user who needs to learn a tool quickly. On-line help can also guide an experienced knowledge engineer through the more complicated tools. Furthermore, the end user may require on-line help in order to properly respond to questions during a consultation.

5.5.1.10 Explanation Facilities

A common feature in many development tools involves some type of conclusion explanation facilities. During a consultation, this feature is used to describe the line of reasoning that the system followed in reaching a conclusion. The developer may also wish to use a consultation tracing feature to aid in debugging the knowledge base.

5.5.2 User Requirements

When selecting a development environment, it is important to take into consideration what the user requirements are. Remember that user acceptance will ultimately make or break your application.

5.5.2.1 Developer Requirements

The level of the developer's computer skills will be a determining factor in the selection of a tool. Some tools are designed for the novice user while others offer complex features that an advanced programmer will utilize. For example, a beginner's tool is usually restricted to a specific control structure with a simple editor to create the knowledge base. However, an advanced user may wish to experiment with various inferencing techniques and knowledge representations. Generally, the more flexible tools have steeper learning curves. For this reason it is recommended that a new

developer start with a more structured shell to gain experience with expert system technology rather than becoming entangled in the details of a complicated development environment.

5.5.2.2 End User Requirements

The end user's requirements should also be considered in tool selection. How well does the user understand the problem domain? How computer literate is the end user? The answers to these questions will determine what level of on-line help and reasoning explanation the final system should offer. Furthermore, the delivery system should be more [robust] if the target user is less experienced. If incorrect or incomplete data is given during a consultation, the system may need to inform a novice of the inconsistencies.

5.5.3 Migration to Delivery Environments

The target delivery environment should be considered early in the system design. First, the available hardware in the user base needs to be determined. It may be necessary for the end users to buy new hardware or to upgrade their current systems. A more cost effective solution could be to choose a shell that is compatible with the user's system. Many tools are designed solely for the PC environment and do not allow portability to a mainframe. Other tools are set up to run only on mainframes or dedicated AI environments. If the expert system is to interact with an existing system, special interfaces may be necessary. Some shells offer a link-up capability with databases and external files with data-passing functions. The more flexible tools allow the user to customize the shell with specialized routines that integrate with other languages.

5.5.4 User Interfaces

Several different types of user interfaces must be taken into consideration during the Definition and Design Phase.

5.5.4.1 Developer Interface

There are a wide variety of developer interface features available from different tools. The ease-of-use of a development environment is a function of the type of features employed in the system. Below are some of the features that can be found in the developer's interface:

- Specialized editors that guide the user through the creation of a knowledge base,
- Decision tree tracing - a graphic or textual listing of questions and responses, and

- Debugging capabilities with a split screen showing the inferencing during a consultation.

5.5.4.2 End User Interface

End user interface features also vary from tool to tool. The features required will depend on the target user as described in Section 4.4.4. Some of the desirable end user interface features include:

- Help features - the "Why" feature can be used during a consultation to have the system explain why the current question is being asked, and the "How" feature can be used after a conclusion has been reached to show the line of reasoning,

- Active images - these can display values and scenarios or allow the user interact pictorially with the system during a consultation, and

- Re-run consultations - these will save the responses in a consultation and allow the user to change a set of parameters to see the outcome in a "What if" situation.

5.5.5 Vendor Information

Information supplied by vendors plays a significant role in the design and eventual development of the expert system. Given below are several areas which must be considered.

5.5.5.1 Stability

An initial consideration in tool selection is the stability of the vendor. A new, small company may have an interesting product, but may not have as an impressive a track record as a more established firm. Keep in mind that all post-purchase customer support is inherently dependent on the vendor's longevity. The following questions should be posed to assess the vendor's stability:

- How long has the vendor been in business?

- Is the vendor in good financial standing?

- How many systems have been installed?

- What types of organizations have purchased the tool?

- How have other organizations used the tool?

- Are there user's groups for the tool?

- Are other users satisfied with the product?
- What is it about this tool that makes it worthwhile to invest in it? and
- What are the risks of developing the application with this tool as compared to all others?

5.5.5.2 Training

Training issues are especially important in the choice of a vendor and shell. Expert system development tools range from easy-to-use yet limited shells to complex, powerful environments. If the latter is best suited for the problem domain, proper training will be an essential contribution to the success of the project. The danger lies in the flexibility of the more powerful tools. Such environments are geared toward research and allow the developer to select from a variety of control structures. Training on these systems is necessary to prevent the user from implementing the wrong knowledge representation or inferencing strategy. Vendors offer varying levels of training, with the most extensive generally reserved for the complex and expensive tools. The types of training available include:

- Training manuals - should be easy to comprehend and geared for the user's level of expertise
- On-line tutorials - can be very useful in exploring the basic features of the tool
- Training courses - offer a deeper understanding of the system's implementation and can be tailored to the user's problem domain
- Consultants - may be needed to assist the developer with complex applications yet are available only through the larger, more established vendors.

5.5.5.3 Cost

Cost is also a deciding factor when choosing a vendor. Often, in addition to the price of the initial development tool, there are other potential costs which should be considered. Listed below are common items associated with expert system development environments whose costs must be considered.

- Initial development system,

- Runtime copies for the end users, or royalties for use on a network,
- Additional AI language interfaces that may be needed to run the development package,
- Specialized AI hardware (e.g. LISP workstations) that is necessary for some of the more powerful tools,
- Update/upgrade for new versions; often the initial cost can be applied toward that of the upgraded system,
- User support provided by the vendor's trouble-shooting hotline, and
- Training, as shown in Section 5.5.5.2 in order of increasing expense.

5.5.6 Documentation

The documentation of the development environment should be both comprehensive and easy to understand. The more powerful development tools generally have extensive documentation. Many tools offer tutorials that guide the developer through example prototypes. A few environments have on-line documentation.

5.5.7 Overall Evaluation

An overall evaluation of the development environment should include all of the items discussed above. Special consideration should be given to the cost of the tool and associated hardware versus the tool's functionality. In comparing various environments, try to rank the features in order of importance for the given application. Be sure to have a well defined problem domain, formalized during the Concept Phase, before attempting to select a development environment.

5.6 DEVELOPER QUALIFICATIONS

In Section 4.12, the task of selecting a project team was initiated. The aspects of project staffing that were emphasized were the size and composition of the project team. This Section will focus on the technical qualifications of the team members.

5.6.1 Knowledge Engineer

The knowledge engineer is responsible for helping to design, build, and validate the expert system. As such, the knowledge engineer should have the following qualifications:

- Good communications skills, particularly in listening, interviewing, and writing,
- Basic computer skills, knowledge of requirements analysis and design techniques,
- Fundamental knowledge of expert systems and artificial intelligence in areas such as knowledge representation and user interfaces, and
- Some expertise in the problem domain.

Desirable qualities also include the ability to work well with people, experience in expert system development, a minimal background in psychology, and experience in systems analysis.

5.6.2 Programmer

The programmer should have fundamental programming skills and experience in the selected expert system development environment. Desirable qualities include familiarity with artificial intelligence techniques, system integration, testing, and validation.

5.6.3 Reviewer/Approver

The reviewer/approver should be familiar with expert systems technology and LCM issues, cost justification techniques, and other forms of information-processing projects.

5.6.4 Expert

The expert should be a recognized leader in the field in which the expert system is being applied. Desirable qualifications include an interest in the project, availability, and good communications skills.

5.7 DESIGN

The design is important to the success of the expert system application as it will be a guide for the Development Stage and the additional prototyping which takes place there. There are two steps to the design process. Each has a specific purpose.

5.7.1 One-page Design

The One-Page Design Document is a very high level view of the system. It is completed at the beginning of the Definition and Design Phase after preliminary knowledge acquisition has taken place. Its purpose is to give a starting point to the design by giving brief description of each of the components of the expert system. It should be emphasized that this document

should be limited to one page, hence the name. This document is broken down into the following six components :

- Overview of the system: This paragraph briefly states the purpose and scope of the system. It also gives an estimate of the measures of success of the system.
- Components and structure: A list of all components or subdomains of knowledge and a brief description of how each fits together in the overall structure.
- Interfaces: A description of any known interfaces including any database and external interfaces. It also states how the user interface will be established.
- External calls and data sources: An enumeration of all external calls the expert system has to make and any data sources which must be utilized by the expert system.
- Knowledge representation: A best approximation as to the appropriate representation, based on what is currently known about the problem domain. It also includes reasons for the selection, because they may be useful later in this phase and on into the Development Stage.
- Inferencing mechanism: The type of inferencing mechanism is determined primarily from the knowledge representation scheme. Refer to Section 4.9 to learn more about how to select an inferencing mechanism.

5.7.2 Detailed Design

The Detailed Design Document further elaborates what was borne out in the One-Page Design Document. In this document, the system is broken out by expert system component or context. Within each component these questions should be answered:

- What is the purpose of the component?
- What is the scope of the component?
- What information is the component going to need?
- Are there any specific external calls and/or data sources needed?

5.8 PROTOTYPING STEPS

The prototyping which began in the Concept Phase continues during this phase with the refinement of the Proof-of-Concept Prototype and the development of the Demonstration Prototype.

5.8.1 Proof-of-Concept Prototype

The proof-of-concept prototype is a small working system designed to provide a preliminary feasibility assessment on the problem domain before additional resources are allocated. This prototype is started in the Concept Phase, but is refined and achieves maturity here. Ideally, this prototype is designed to include only the top level features of the targeted system. For example, the goal recommendations of the knowledge base can be represented without the full line of reasoning to handle all possible cases in the problem domain. The Detailed Design Document is merely alluded to and need not be included at this stage.

5.8.2 Demonstration Prototype

The Demonstration Prototype is an extension of the Proof-of-Concept Prototype. The Demonstration Prototype should be small and specialized, based on a narrow subset of the overall problem domain. In contrast to the Proof-of-Concept Prototype, an essential element is that the knowledge base is designed to be deep in one area while maintaining breadth in the other areas. In this prototype, more attention is paid to user interface features to make the system attractive to management and to the end users. Also, the necessary data requirements are addressed at this stage to enable the assessment and design of the external data integration.

CHAPTER 6
DEVELOPMENT STAGE

6.0 INTRODUCTION

The Development Stage is a translation of the design into a working system. This chapter describes several success factors needed during development, the types of prototypes produced, and knowledge acquisition.

6.1 OBJECTIVES

There are several key objectives in the Development Stage (Fig. 6.1). One is to document key issues to be borne out during prototyping. These issues are discussed for each prototype in Section 6.10. In the Development Stage key issues for the Implementation Stage should be documented. The problem description, functional description and knowledge are refined during development of the prototypes. Now is the time to increase product visibility. Also, maintaining management commitment is a key objective of the Development Stage. Another objective is to codify the knowledge base and maintain its accuracy through testing.

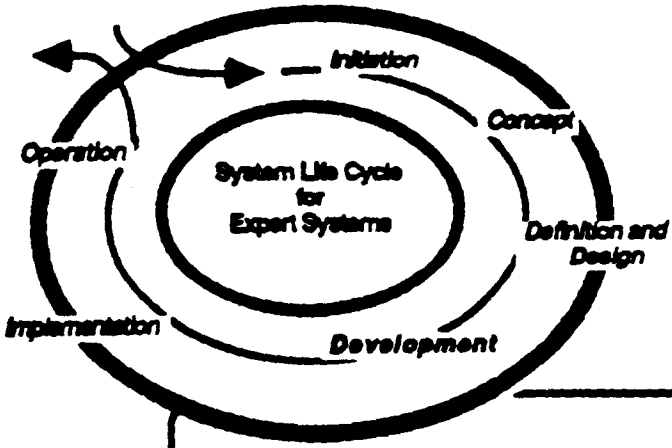
6.2 DECISIONS

There are several key decisions that must be made during the Development Stage (Fig. 6.1). The first decision is how to proceed given current funding levels. Before each prototype is begun, two questions must be resolved. The first is, "To what degree has knowledge acquisition taken place?" The second question is, "Is the prototype addressing the problem sufficiently to be fully developed?" User and management commitment are very important to the success of the project so decisions must be made to satisfy both user and managerial requirements. Another decision that must be addressed is whether the developer has identified and resolved security and backup issues.

6.3 PRODUCTS

There are several products associated with the Development Stage (Fig. 6.1).

- Project management plan
- Test and validation plan
- Support plan
- Knowledge notebook
- Full Prototype
- Production System
- Development documentation
- Development decision paper.



Objectives

- Document key issues to be borne out during prototyping
- Document key decisions for the implementation phase
- Perform implementation planning before full implementation
- Refine knowledge
- Refine the problem description and functional description
- Obtain user acceptance
- Increase product visibility
- Maintain management commitment
- Codify knowledge
- Maintain knowledge accuracy through testing

Decisions

- Decisons to be addressed include:
  - how should development activities proceed given current funding levels
  - has the prototype addressed the problem sufficiently to be implemented
  - to what degree has knowledge aquisition taken place
  - has the project gained user and management commitment
  - have security and backup issues been identified and resolved

New Products

- Project Management Plan
- Test and Validation Plan
- Support Plan
- Knowledge Notebook
- Full Prototype(s) (if needed)
- Production System
- Development Documentation
- Development Decision Paper

Figure 6.1

Development Stage
Objectives, Decisions, and Products

6.3.1 Project Management Plan

The Project Management Plan is used to make sure the Development Stage is run smoothly. It specifies who is in charge of each step of development. It also specifies the resources each person has control over and outlines the hierarchy of management when there are conflicts in the decision making process.

6.3.2 Test and Validation Plan

The Test and Validation Plan is developed to specify how testing and validation should be handled. It will contain a methodology for designing tests for the expert systems, and it will list resources available during testing as well as specifying when testing and validation will take place. Key staff involved in the validation process are named at this time.

6.3.3 Support Plan

The Support Plan contains information on what support services are available. All resources and the time they are needed are specified in this plan. Refer to Section 2.5 for information on what resources are necessary in the Development Stage.

6.3.4 Knowledge Notebook

The Knowledge Notebook is used throughout the Development Stage for knowledge acquisition and is kept by the primary knowledge engineer. Refer to Section 6.9 for a description of this document.

6.3.5 Full prototype

The Full Prototype is an intermediate product in the Development Stage. This prototype is described in Section 6.10.1.

6.3.6 Production System

The Production System is the primary product of this stage. This stage is not complete until the Production System is produced. The rest this chapter outlines the steps necessary to produce it.

6.3.7 Development Documentation

The Development Documentation is produced while the prototypes are being developed. It contains information on how each prototype was developed and documentation on the knowledge representation and control structures. It also contains the documentation for any supporting software of the expert system.

6.3.8 Development Decision Paper

The Development Decision Paper is a documentation of all decisions that were made during this stage. It lists each decision and the reasons and facts supporting why the decision was made.

6.4 SUCCESS FACTORS

There are several areas that are crucial to the success of the Development Stage. These factors are grouped below according to the area they address.

6.4.1 Knowledge Engineering

Knowledge engineering begins in the Concept Phase. The knowledge engineers should devise a method to resolve conflicts among multiple sources of expertise with differing specialties. Methods for dealing with conflicts are given in Section 6.8.

It is wise to coordinate programmers and knowledge engineers, but keep in mind that there may be organizational separation and differences in background. Knowledge engineers should have a strong computer background to facilitate communication with programmers. Knowledge engineering is most effective when proven knowledge acquisition methods are used. These include :

- Unstructured interviews,
- Structured interviews,
- Observation,
- Interruption analysis,
- Constrained-processing tasks,
- Questionnaires, and
- Decision trees and decision tables.

Many good techniques are available. The knowledge engineer should apply one or several structured and unstructured techniques to document the expert's [domain knowledge]. These techniques are discussed in detail in Section 6.9.2.

The knowledge engineers and expert should think out all of the ramifications of the rules on a periodic basis. There should be development of knowledge throughout the project to allow for constant refinement and experimentation. It is a good idea to periodically recheck the accumulated knowledge in order to better refine it.

Knowledge engineers need not be experts in the field; too much domain knowledge has a risk of producing biases in the process toward the knowledge engineers and away from users. Select knowledge engineers who are familiar with the domain, but are not necessarily expert in it.

Knowledge engineering time estimates should be well thought out with respect to reaching the appropriate depth of knowledge. There should be a constant reminder to the project manager to plan for contingencies and expect repeat sessions with experts and users.

6.4.2 Prototyping Methodologies

Below are several methods that contribute to a successful prototype.

- Use rapid prototyping with frequent interim deliverables and decision points.
- Discard the prototype if a better design approach is discovered. The purpose of a prototype is to firm up the design specifications.
- The design should be modular to show all lines of reasoning and specific functions the expert system is required to perform. The prototype should also be attractive as a demonstration vehicle to gain the support of users and management. The development team should accurately define and adhere to the system development stages. This will allow the team to anticipate the long-term impact of schedule deviations.

6.4.3 Validation Process

It is important to identify and validate external interfaces as early as possible. This includes inputs and outputs, and other programs, and algorithms - separately from the knowledge bases. Validation should be done frequently and continually. There is a tendency to ignore validation until the end of the Development Stage. Ideally, it should be done after the addition of each rule, but later can be reduced to the end of each session. Expert system validity relies heavily on the validity of accessed data. Issues to be considered when validating include the knowledge base, recommendations, justification, rationale, type of inferencing and incomplete data.

It is not always possible to test all possible rule outcomes. This is often the case for larger systems, e.g, +500 rules. This emphasizes the importance of keeping a "trace" of each session and the need for software maintenance in the operational phase.

6.4.4 Testing Procedures

Testing is the most important part of the Development Stage because there may be potential conflicts in the knowledge base. A thorough analysis should be performed on a routine basis throughout development and upon completion of development. Remember to test all possible rule combinations to identify conflicting or overlapping rules. Some ways to do this are :

- o Random tests by beta users,
- o Selected test cases run through the system, and
- o Ad-hoc testing by the expert.

Make sure to adequately test critical components and any examples or rules generated by inductive systems. Thoroughly retest entire knowledge base when changes are made. The process of testing expert system shells is easier than testing implementations written in LISP or PROLOG because the inference engines have already been thoroughly tested in a shell. Remember not to underestimate the proper level of testing. Allocate sufficient time and resources to do the testing.

6.5 ACTIVITIES

Two background activities should be completed during this stage to ensure a smooth development of the expert system. These activities include the end users and management staff.

6.5.1 Demonstrations and Briefings to Users

Periodically, and throughout the Development Stage, conduct demonstrations for the users. In addition, the user should play a primary role in the development of the user interface in order to assure acceptance of the expert system in its delivery environment. User involvement here will ensure ongoing support for the expert system application. Frequent demonstrations of the system and milestone reports to management for the expert system application are necessary for continued user involvement.

6.5.2 On-going Progress Reporting to Management

Management should be updated on a continual basis as to the progress of the development effort. If time or cost overruns are expected, management should be given a detailed report. Using detailed progress reports, it is advisable to demonstrate to management that as the project changes and the requirements are refined, the effort continues to be manageable and important.

On-going management commitment is vital to the success of the expert system. To ensure management commitment, detailed progress reports should be given weekly or in an appropriate time frame for the project. Delays should be attributed to discrete events and not to the expert system technology itself. If proper feedback is given, management will be more confident that the development effort is under control.

6.6 KNOWLEDGE ACQUISITION

Knowledge acquisition is the process of extracting and representing the expert's knowledge in the form of a conceptual model. A point which is sometimes overlooked in acquiring knowledge is the fact that there are usually multiple solution paths to reaching a conclusion, based on alternative hierarchies of assumptions. Expert systems must be designed to deal with these types of uncertainties, which vary depending on the domain being examined. Experts are not always explicitly aware of precise concepts and representations of their knowledge. Documenting this knowledge requires considerable patience and cooperation between the knowledge engineers and the expert. In order to conceptualize the problem the expert and knowledge engineer must agree on issues such as:

- What are the factors involved in decision making?
- Which inputs give the expert difficulty?
- What are some of the relationships between factors?
- What factors are missing?
- How accurate are the factors?
- What inferences does the expert make?
- How are hypotheses formed?
- How does the expert's knowledge evolve?
- What factors suggest particular goals or concepts?
- What are the solutions?

6.6.1 Knowledge Sources

Several different [knowledge sources] the knowledge engineer can draw upon while studying the problem are outlined below. Sometimes conflicts in data will emerge from different sources.

6.6.1.1 Background Data

The knowledge engineer must be familiar with the domain so as not to overly burden the expert with questions that can be answered from background reading. Background can come from various sources including regulations, guidelines, textbooks, and deliverables or talking with the expert. Any policies and/or procedures that are already outlined or documented can be a valuable source of knowledge.

6.6.1.2 Cases

By analyzing specific cases, the knowledge engineer will be able to determine what information is potentially important and how the factors are interrelated. Cases also give the knowledge engineer an overall feel for how the problems were solved in the past.

6.6.1.3 Test Data

Providing test data or posing hypothetical situations to the expert is another way the knowledge engineer may extract information. This is done by observing the way the expert solves the problem with data provided in a structured test format. Test data can be derived from historical data allowing dummy cases to be contrived for purposes of gaining knowledge.

6.7 COLLECTION METHODS

Given below are various methods for acquiring knowledge about the domain. First though, are some general interviewing principles.

6.7.1 Interviewing Principles

Points to remember when interviewing an expert include the following:

- The knowledge engineer should be specific when asking questions. The expert may not have remembered rules or concepts, and will find it difficult to recall them.
- The expert should be encouraged to provide the information in a way which is most natural.
- The expert should not be interrupted during unstructured interviews. The aim is to get the expert talking. Despite the fact that the expert will probably digress or repeat things, interruptions should be kept to a minimum.
- It is important for the knowledge engineer to record all information collected during an interview and save it in the *Knowledge Notebook*. It is not always clear which parts of the dialogue are important, even if the questions are planned. The use of a tape recorder or video can be very useful if the expert is comfortable having the information acquired using these techniques.
- The knowledge engineer should listen to the way the expert uses knowledge. It is not just facts, theories, and heuristic that are important. The knowledge engineer should listen carefully to the way in which the expert manipulates knowledge.

6.7.2 Knowledge Acquisition Methodologies

There are various proven knowledge acquisition techniques which currently exist. A few are described below.

6.7.2.1 Unstructured Interview

An unstructured interview is basically a free form interview in which the expert talks freely about the domain of expertise. The goal is to become comfortable with the expert and to see how the expert views the problem domain. The unstructured interview is not very efficient, but this technique can be used to address several important questions:

- Who are the experts?
- Is the problem scope suitable for an expert system?
- What are the needs of the end users?

6.7.2.2 Structured Interview

The structured interview is the most basic knowledge collection technique used by the knowledge engineer. It is generally used after unstructured interviews are conducted. This method is applicable to all types of problems and with all types of experts. After the knowledge engineer has met with the expert several times and structured some of the knowledge, the knowledge engineer then tailors the interview around gaps in the information. Interview questions should center on the expert's knowledge of factors, relationships, and inferences.

6.7.2.3 Observation

In some cases, the best way to discover how an expert makes a judgment, diagnosis, or design decision is to watch the expert work. This method is used primarily when the task at hand involves more than just a cognitive process to achieve a goal. For instance, determining which equations the expert refers to most often while solving a problem. The knowledge engineer may discover the knowledge in various ways.

One way is to watch, take notes and follow the expert's thinking processes. The drawbacks to this are that the process is time consuming and that the expert may perform differently while being observed.

A second method is to record the session. This can be advantageous because it:

- Retains all information from the interview,
- Can keep pace with the expert, and
- Keeps the knowledge engineer's attention on the interview and not on recording the information.

The drawbacks to this method are:

- All information must be transcribed,
- Irrelevant pieces must be filtered out,
- There is nothing to refer to during the interview, and
- The expert may be intimidated by a recording device.

6.7.2.4 Decision Matrix

Here the knowledge engineer forms a table or matrix with the decisions to be made along one axis and the different factors that go into making these decisions on the other. This method is used when many factors are borne out during interviewing and must be organized in some fashion. The purpose is to better diagram the decision making process and to identify gaps in the knowledge for future interviews.

6.7.2.5 Interruption Analysis

Interruption Analysis is especially good for validation of the reasoning processes already encoded in the expert system application. It works in conjunction with the observation technique described in 6.7.2.3. In this method the knowledge engineer observes while the expert proceeds to solve a problem without verbalizing the process. When the process gets to the point that the observer can no longer understand the expert's actions, the knowledge engineer interrupts. The knowledge engineer then probes deeper, asking the expert about what was happening at that particular instant. Note that once the process is interrupted there is a chance that it cannot be started up again.

6.7.2.6 Constrained-Processing Tasks

In this method the expert is requested to solve the problem in a much shortened time frame or with limited information. Experts are generally uncomfortable with limited information or time available to solve the problem. Some experts would rather give no answer at all than give one based on incomplete information. Explain to the expert that this is not a test of their expertise but rather a means

of extracting additional information about the domain. Once the expert opens up and becomes more comfortable about giving uncertain or qualified judgments, much can be learned about the experts's reasoning processes. There are two types of constrained processing tasks.

The first is a constrained time task. Constrained time tasks limit the time an expert has to solve the problem. When a time constraint is imposed, the expert will generally take the most direct path to the solution. This method is used to eliminate extraneous thought processes which may be occurring without the expert realizing them.

The second is to limit the information available to the expert. This is a good way to determine which factors or data are most significant by forcing the expert to rely heavily on their knowledge and reasoning skills. Formulation of hypothesis, use of heuristic, and strategic thinking are important in limited information tasks. Often, the expert is unaware of factors which are most important and this is an excellent way of finding out.

6.8 RESOLVING CONFLICTS

As the knowledge engineer gains more and more knowledge about the domain, conflicting information builds. Listed below are some techniques used to resolve these conflicts.

6.8.1 Focus Groups

When there are cases where multiple experts having different domains of knowledge are involved, focus groups can be formed to decide on the best method of resolving the conflict.

6.8.2 Delphi Method

The delphi method consists of writing out each piece of conflicting information and evaluating each for its own merit without regard as to its source. Rationale is given for each and an agreement is made as to the best answer.

6.8.3 Consensus

A general consensus can be made based on discussions with multiple experts. The disagreement may have been simply the way in which the knowledge was represented or presented This method is often best when dealing with incomplete or inconsistent knowledge.

6.8.4 Quality Assurance Committee

The quality assurance committee is set up specifically to resolve conflicts and maintain knowledge integrity. This committee should be set up initially if many conflicts are expected to occur among experts.

6.8.5 Hierarchically

Though not the ideal way, sometimes it is more efficient to resolve conflicts in a hierarchical fashion. If people at one level have trouble resolving a conflict the person higher up in the chain of command such as a department chief or manager will resolve it. This method works best early in the Development Stage where decisions must be made quickly to avoid unnecessary delays. If the manager being asked to make the decision is not a domain expert, reconsider escalating the issue to that level.

6.9 KNOWLEDGE NOTEBOOK

The *Knowledge Notebook* contains notes on all knowledge engineering sessions with the expert including any knowledge maps the knowledge engineer has developed and any real or perceived knowledge gaps. The Knowledge Notebook serves to organize the knowledge engineer knowledge and prevent the knowledge engineer from asking the expert redundant information.

6.10 PROTOTYPING STEPS

Prototyping reaches its final stages during the Development Stage. Below are descriptions and development methodologies of the Full Prototype and the Production System produced during the Development Stage.

6.10.1 Full Prototype

The Full Prototype contains most of the knowledge and the representation and control structures have been determined. Sample cases and lines of reasoning are expanded upon. Difficult cases are added and any external interfaces are connected. Testing and verification now play a more important role as information and lines of reasoning are expanded.

The purpose of the Full Prototype is to verify the problem domain, the knowledge representation schemes and control structures, data requirements, and interfaces. If major revisions are needed either to the knowledge representation or control strategies, a new approach can be designed following the guidelines in CHAPTER 4.

Prototyping is an exploratory process. If at this point the design is no longer valid, the prototype can be reevaluated based on the new information and redesigned if appropriate. The efforts that went into the

development are not lost; they represent the design phase in conventional systems.

6.10.2 Production System

This is the actual system that will go out into the field. At this point development is complete and the expert system has a finished knowledge base which has been compiled and is secure. Any additions or deletions which must be performed as a result of further evaluation and testing are implemented. All user interfaces and integration with external interfaces are in place. In addition, the system has been fully tested and validated. Any disclaimers about the use and liability of the expert system are added to this prototype. The completed system should be robust and user ready.

6.11 TESTING AND VALIDATION

Prior to the field testing performed in the Implementation Stage (see Section 7.6.2), it is important that the expert system is tested and validated by the developers. This will ensure that the Production System is working effectively. The items to be tested include the data, the knowledge base, and the flow of logic.

6.11.1 Data

Data validation is the most often neglected task in expert system development. Databases, external files, and tables must be correct before the expert can reason using this data. If the data is incorrect the system will fail to produce the required results. This is true of the results to date as well as any future results.

6.11.2 Knowledge Base

The data that results from the knowledge acquisition processes should be comprised only of valid information. The data should contain truths about the expert's knowledge and reasoning and the expert's perceptions and observations on the domain. Also, the data must be complete in the sense that it covers all of the relevant subdomains of knowledge.

The knowledge should accurately reflect the way the expert views the domain and reasons about the task at hand. One way to check this is to see if there is a consensus across experts. Remember that some people feel a bias toward a computer generated result and may grade the response more harshly than one would an expert. To get around this, intermix the computer results with those of an expert panel without differentiating between the two.

6.11.3 Logic Flow

There are various methods used to validate the logic flow.

- Run test cases using sample problems to see if the expert system arrives at the same conclusion the expert does.
- Follow chains of reasoning, picking one specific topic within the domain and following the reasoning process the expert system follows to see if it matches the reasoning followed by the expert.
- Change the firing sequence to see if the expert system produces the same results if the firing sequence is changed. If the system fails to respond in the same manner, then the result was a artifact of the control structure and not of the knowledge.

CHAPTER 7
IMPLEMENTATION STAGE

7.0 INTRODUCTION

The Implementation Stage is the second part of the Development and Implementation Phase. It is the last step before the system is made fully operational to all of the users in the field. At this point, the prototype system has been tested by the expert and the developer, and revisions have been made. The system, in its nearly final state, is ready to be tested in the field by the actual users through the beta test. The Implementation Stage includes four main steps:

- Testing and evaluation of the expert system by beta test participants
- Revising the expert system by knowledge engineers
- Registering users and distributing the system
- Training users and providing System Documentation.

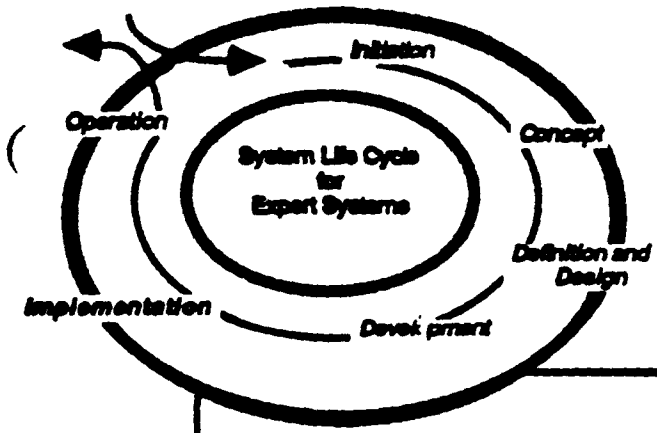
7.1 OBJECTIVES

The objectives of the Implementation Stage (Fig. 7.1) are mainly to test and finalize the expert system and prepare it for distribution. This includes a beta test of the system by potential users and a revision of the system following the test. In preparation for distribution of the system, a training program and documentation are completed, and users are registered. The expert system itself is also prepared for distribution through final debugging and copies of the run-time version are produced.

7.2 DECISIONS

A number of important decisions are made during the Implementation Stage (Fig. 7.1). These involve the testing and revision of the system, as well as the distribution of the system and training of users. Decisions involving the beta test include determining who will participate in the test, and how long the test will last. The group of beta users should be kept to a relatively small number in order to limit the volume of comments on the system. However, the beta users should comprise a representative sampling of the types of potential users of the system. An intensive beta test should generally last 2-3 weeks, or longer if necessary. This will give the beta users adequate time to learn how to operate the system, apply it to their work, and evaluate it.

After beta users comments have been received, the developers of the expert system must decide what revisions they will make. This will involve reviewing the comments and determining how critical the suggested changes are



Objectives

- Have beta users test expert system in the field
- Revise the expert system if necessary
- Prepare training and documentation for the system
- Prepare system for full distribution

Decisions

- Decisions to be addressed include:
  - who will test the expert system
  - how long will the beta test last
  - what revisions will be made to the system
  - is the expert system ready for distribution
  - how will training be implemented
  - is sufficient funding available for the remainder of the life cycle

New Products

- List of Beta Test Participants
- Compilation of Beta User Comments
- Scope of User Community
- Documentation of Expert System
- Distribution Plan and Schedule
- Implementation Decision Paper

Figure 7.1

Implementation Stage
Objectives, Decisions, and Products

to the success of the expert system. Project management must agree to the developer's recommendations of what revisions to make. There is an important trade-off between achieving functionality in the system and using resources efficiently, and this should be considered when determining which revisions are the most important. As revisions are implemented, the developers will decide when the system is ready for final distribution.

Decisions involving the implementation of a training program are also made at this time. Training options are discussed in Section 7.6.4. As this stage in the life cycle comes to a close, management staff must determine that there is sufficient funding for the remaining phases.

7.3 PRODUCTS

The Implementation Stage results in six products (Fig. 7.1). These include:

- Distribution Plan and Schedule
- Implementation Decision Paper
- Scope of User Community
- List of Beta Test Participants
- System Documentation
- Beta User Comments

7.3.1 Distribution Plan and Schedule

At the beginning of this stage, it is important to draw up a Distribution Plan and Schedule, summarizing the distribution activities to be completed. Following the schedule closely is especially critical in this stage because of the increased contact with people outside of the system development staff.

7.3.2 Implementation Decision Paper

The Implementation Decision Paper should be composed early in this stage. It should include all of the decisions to be made during this stage (see Section 7.2), and who is responsible for carrying them out. The Implementation Decision Paper should be approved by the project management staff to ensure that they are aware of the decisions and schedule that must be carried out.

7.3.3 Scope of the User Community

The final Scope of the User Community must be established at this time in order to plan for how many systems will be distributed and to provide adequate support once the system is shipped to the field.

7.3.4 List of Beta Test Participants

Potential beta test participants are first selected from the group of end users. Beta test participants should include those who are interested in the development of the system and have the time to effectively test the system. After contacting the potential beta test participants and locating those willing to be involved, a List of Beta Test Participants will be compiled.

7.3.5 System Documentation

Documentation of the expert system and instructions for its use are written during this stage. It is important that these documents are completed, at least in draft form, prior to the beta test, in order that they can also be evaluated. Further requirements for System Documentation are covered in Sections 7.4.1 and 7.6.5.

7.3.6 Beta User Comments

Following the beta test, the beta test participants will be contacted and their comments on the system will be solicited. These comments will be integrated and compiled for use by those involved in the revision of the expert system.

7.4 SUCCESS FACTORS

There are several factors in the Implementation Stage that must be considered in order to ensure the success of the expert system. Most of the factors involve the transfer of the system to users in the field and the initial use of the system. In order to produce a successful expert system, it is important to:

- Provide useful, readable documentation
- Provide sufficient and organized training
- Provide sufficient technical support
- Ensure that hardware in the field is adequate to support the expert system software
- Avoid problems with licensing and run-time versions
- Maintain management commitment during the distribution of the system.

Most of these factors can be covered through planning and coordination prior to this stage.

7.4.1 Documentation

Good documentation can be produced through a quality assurance system, requiring reviews by the developers of the system, as well as management. A draft of the documentation must be provided to the beta users for their comments on its clarity and comprehensiveness. For more information on expert system documentation see Section 7.6.5.

7.4.2 Training

The planning of the training process should be supervised by management, who will work with the developers and the trainers. For more information on training issues, see Section 7.6.4.

7.4.3 Technical Support

In order to provide sufficient technical support, a technical support team should be organized before the system is distributed. Their duties should include manning a support hotline and providing on-line help to the users. The technical support team should include individuals who were involved in developing the expert system, writing the documentation, and providing training sessions. More information on technical support for the expert system can be found in Section 8.5.2.2.

7.4.4 Hardware Issues

Problems with hardware in the field can be avoided if the users are given ample warning of what equipment will be required to run the expert system. This process is explained in Section 7.6.6.

7.4.5 Licensing Issues

Licensing and run-time issues should have been confronted early in the life cycle (see Section 2.5.6.1). During the Implementation Stage, it should be necessary only to confirm with the vendor of the shell what steps should be taken to distribute run-time copies of the expert system. The matter of who will pay for the run-time copies must also be considered at this time. For more information on run-time and licensing issues, see Section 7.6.8.

7.4.6 Role of Management

Management commitment during this stage can be ensured if, early in the life cycle, management is made aware of the fact that they will play a large role in the Implementation Stage. They should know that they will be responsible for the scheduling and planning involved in this stage, as well as the actual distribution of the system and all of the issues accompanying the distribution process.

7.5 ACTIVITIES

There are several important activities in this stage that lead up to the distribution of the system. These include the collection of data, testing, demonstrations, and management activities.

7.5.1 Data Collection

Collection and organization of data is important during this stage. Sufficient staff time should be allocated for the following tasks:

- Contacting possible users and recording their level of interest in participating in the beta test
- Selecting actual beta users and recording their names and addresses on a mailing list
- Seeing that the beta test system and any accompanying documentation is sent to all of the beta users
- Developing a questionnaire on the system and sending it to all of the beta users or conducting the survey by phone
- Collecting the questionnaires and compiling responses and suggestions from the beta users
- Registering all users of the system and maintaining the registry.

7.5.2 Testing

The main testing to be conducted during this stage will be the beta test. Resources should be allocated for the following tasks related to the beta test:

- Contacting beta users (see Section 7.5.1) before and after the test
- Technical support for both hardware and software
- Documentation/training
- Revisions to the system following the beta test.

7.5.3 Management Support

Management plays an important role in the Implementation Stage, so it may be advisable to allocate a staff member to serve as an assistant to the manager. This person would provide administrative support for:

- The beta test,
- Data collection,
- Training and documentation,
- Licensing and run-time arrangements,
- Distribution plans, and
- Configuration management.

7.5.4 Demonstrations and Briefings for the Users

Contact with the users is especially important during the Implementation Stage, as the expert system is nearing its final state. The users will be able to see the actual system they will be working with, rather than a prototype. Demonstrations and briefings involving possible users can be useful in determining who is interested in participating in the beta test and final evaluation of the system. They will also determine the overall level of interest in the system. This information can be used to define the size of the system, and to allocate sufficient staff time for distribution, technical support, and maintenance of the system.

Demonstrations and briefings can also be used as training sessions for both the beta users and initial system users. In some cases, the developers of the system may choose to meet with some of the beta users after the test to receive their comments on the system. This would allow for a more interactive evaluation than simply filling out a questionnaire.

7.5.5 On-going Progress Reporting to Management

Management should receive a complete schedule of all tasks involved in implementation at the beginning of the stage. They should immediately be informed of any changes in the schedule, because many aspects of system implementation involve outside parties, and maintaining good public relations is very important. Progress reports should include updates on:

- Beta test,
- Distribution plans, and
- Any major problems encountered.

7.6 DISTRIBUTION STRATEGY

An important part of the Implementation Stage is planning and scheduling the distribution of the system. This stage of implementation has a number of parts and should include close management involvement.

7.6.1 Rollout Plans and Schedule

At the beginning of the Implementation Stage, all tasks to be performed must be carefully planned and scheduled. Specific dates should be set for the beginning and end of the beta test, and for the distribution of

the final system. All aspects of the stage should be planned carefully, and sufficient staff and time should be allocated for each task. It is very important that the beta test and distribution of the final system are carried out smoothly and that the schedule is closely followed, because this stage will be receiving a great deal of external attention, and public relations is an important factor.

7.6.2 Beta Test

The beta test is a very important step in the life cycle of an expert system. During this step, the system will be distributed to a few selected participants who have agreed to try the system out for a defined period of time, then provide their comments on the system and any suggestions they have for changes or improvements. The aspects of the system to be tested include:

- Hardware,
- Software,
- Support procedures,
- Documentation, and
- Training.

The beta test provides an opportunity for the system to be tested on [real-world problems] by the people who will be using the system. They will know best if the system will be effective and assist them in their work.

7.6.2.1 Contact Beta Users

Potential beta users should be contacted early in the Implementation Stage to determine their interest in participating in the test. A small number of interested parties will be selected to participate in the beta test. These participants will receive the entire expert system, along with draft documentation on the system. The beta users can also be provided with training on the use of the system, if necessary. This would provide an opportunity for the training materials to be tested, as well.

7.6.2.2 Register Beta Users

The beta users should be registered before they begin testing the software. This will make it easier to identify unregistered beta users and explain to them the risks of using beta test software. The distributors' risk and exposure will then be limited if the unregistered persons decide to use the system anyway.

7.6.2.3 Collect Comments from Beta Users

After approximately 2-3 weeks, or longer if necessary, the beta users will be required to answer some questions about the expert system and provide comments and suggestions for changes or

improvements. The beta users may be sent a questionnaire, which they will have to complete and return, or they may be contacted by phone and asked the questions directly. The developers may also wish to meet with some or all of the beta users and conduct an open forum to discuss the expert system.

7.6.2.4 Revise Expert System

After the beta users responses and comments have been compiled and evaluated, the developers will revise the expert system accordingly and prepare a final version for distribution to all users.

7.6.3 Registered Users

Registration involves keeping a list of all authorized system users and assigning each with a registration number. This process will be supervised by the management support staff. All users of the final expert system should be registered. However, if there will only be a small number of users, this may not be necessary.

Registration ensures that all of the users are using the official run-time version and that unauthorized users will not be supported. Each user will be assigned a registration number, which will be activated when the user returns the registration card included with the software. The registration number must be stated before any information or technical support will be provided to the user.

A list of all registered users and their addresses and phone numbers will be maintained so that they can be informed of any errors in the system or documentation, and be provided with new versions of the system when they are developed (see Section 7.6.9).

7.6.4 Training Issues

Training of all system users is an important part of ensuring that the expert system is operated and maintained properly in the field. Users can be trained directly through face-to-face training sessions, or indirectly through other training methods.

7.6.4.1 Direct Training

Direct training of the users can reduce the amount of technical support necessary and can make the transition into the initial operation of the expert system much smoother. However, direct training may not be necessary or desirable, if, for example:

- The users are widely scattered geographically,
- There is a large number of users,

- The system is self-explanatory, and/or
- The majority of the users have seen demonstrations of the expert system and have a good understanding of its operation.

If direct training is implemented, those performing the training sessions should be persons who were directly involved in the development process, such as the knowledge engineers, or instructors who have been trained directly by them. Training sessions could be at the users' location, or in a central location, with one or more sessions.

It should be determined if every user will be involved in the training, or if there will be a single representative or small group from each area or region. Training a small number of representatives is known as a "train-the-trainers" approach. After these individuals have been trained, they will be able to train the other users in their own area or region.

7.6.4.2 Other Training Methods

If direct training is not implemented, several other methods of training can be used. Training can be put on-line as part of the expert system, it can be recorded on videotape and sent to the users, or it could be provided in the form of a written training manual.

It may be useful to provide training at several levels. The initial set of training could be a beginning level, and could be followed by more advanced training, after users have some experience with the system.

7.6.4.3 Intended Purpose of the Expert System

An important part of the training will be emphasizing the intended use of the expert system. An expert system is meant to be advisory in nature, and will not take the place of a human. The users will receive recommendations from the expert system, but it remains their responsibility to make all final decisions, either accepting or rejecting the system's recommendations.

7.6.5 Documentation

Documentation of the expert system is very important and is something that is often inadequate. The more clear and easily understandable the documentation is, the less technical support will be needed. The documentation should probably not be written by someone who was directly involved in the development of the expert system, such as a knowledge engineer

or programmer. They may leave out details that are obvious to them but not a first time user.

The developers of the system should be involved in a review of the documentation, but it may be preferable to have the documentation written by someone who is trained by the developers, but who has just become familiar with the system. Such a person will be more aware of every necessary step and detail involved in running the system. Another option would be to use a technical writer, who could study the system and be briefed by its developers, then write the documentation.

The documentation should be brief, clear, concise, and use graphics and pictures of the actual screens whenever possible. No system will be shipped without adequate documentation.

7.6.6 Hardware Requirements

Prior to distribution of the final expert system, all users will need to have the hardware necessary to support the expert system. Early in the Implementation Stage, all potential users should be informed of the hardware, equipment, and facilities they will need to run the system, such as:

- Type of computer, amount of memory, whether a hard disk is necessary
- Type of furniture and work space needed to accommodate the computer
- Type of printers
- Phone lines and modems
- Electrical hookups and surge suppressor.

Ideally, the hardware selection process, described in Section 4.11 took into account the facilities of the majority of the potential users, and the target hardware was chosen accordingly.

7.6.7 Software Requirements

Prior to distribution of the final expert system, the software must be prepared for the users. A run-time copy must be prepared for each user and labeled. Everything necessary for the system should be packaged together and sent to all users with detailed instructions.

7.6.8 Run-time and Licensing Issues

Before the final expert system is distributed, all run-time and licensing issues must be resolved. Although the major decision point on this subject occurred when the development and delivery environments were evaluated

in the Definition and Design Phase (Section 5.5), these issues must be addressed in the Implementation Stage. All user facilities will receive a run-time version of the software. Some of the expert system shells currently on the market require a fee for each run-time copy issued, while others have a one time fee that covers all run-time copies. If the run-time fees are to be allocated to the users, a system for the collection of fees must be implemented.

Some expert system shells do not include run-time modules. These shells do not provide any protection for the expert system, and would allow the users to make changes to it. If this is the case, a disclaimer must be included, stating that the system will no longer be valid if it is tampered with or changed in any way.

The final version of the expert system should include the copyright information for the company that owns the rights to the expert system shell, as well as a notification that it is illegal to make copies of the software.

7.6.9 Configuration Management and Version Control

When the expert system is revised or a new version is developed, it will be necessary to distribute it to all users. Users must have the currently valid version, and only that version, to avoid confusion and legal infractions of the licensing agreement. When a new version becomes available, all users will be notified. They will be required to send in their old version, which is marked with their registration number, in order to receive the new one. Other matters to be considered when the system is revised include:

- Changes or addendum to the documentation
- Retraining
- Removal or overwriting of old versions from all of the users' hard disks.

CHAPTER 8
OPERATION PHASE

8.0 INTRODUCTION

After completing all four major phases (Initiation, Concept, Definition and Design, and Development and Implementation), Operation is the last of the five major phases in the OSWER system life cycle. This phase is divided into three stages, Production, Evaluation, and Archive.

The Production Stage starts with the results of the work conducted in all prior stages, and activities of this stage are frequently in response to those results. In other words, deficiencies and problems not resolved adequately in prior stages are usually identified by users during the Production Stage.

The Evaluation Stage differs from other phases and stages of the system life cycle in that it occurs simultaneously with another stage (Production) and may be repeated more than once during the system's operational life. During this stage, the system is reviewed formally to determine whether it is operating correctly and efficiently from a technical standpoint, and whether it continues to effectively address the information management problem.

The Archive Stage preserves information not only about the current production system, but also about the evolution of the system through its life cycle.

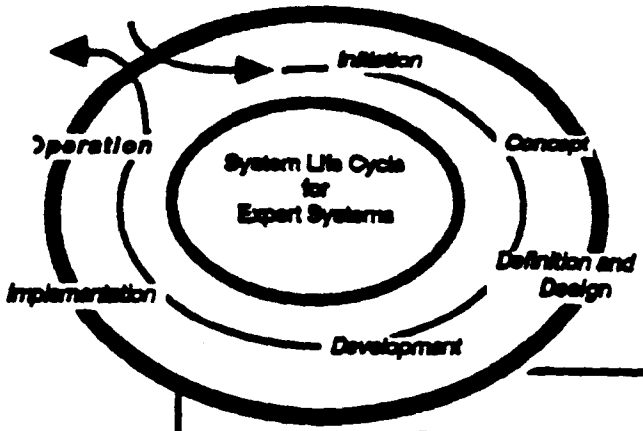
8.1 OBJECTIVES

Figure 8.1 graphically illustrates the objectives associated with the Operation Phase. Each stage's objectives are cited below.

8.1.1 Production

The first objective of the Production Stage is to use the capabilities of the system to solve the information management problem by delivering the system to the users. Proper use of the system will require user training in the special capabilities and limitations of expert systems.

The second objective is to identify potential changes needed to ensure that the system and data continue to solve the information management problem. Changes may take the form of routine maintenance or may constitute enhancements to the system or databases. Careful management of the knowledge base, including revalidation after each change, will be required to ensure continued satisfactory performance.



Objectives

- **Production:**
  - deliver the system to the users
  - identify potential changes to the system
  - develop and implement maintenance changes and minor enhancements
- **Evaluation:**
  - determine the extent to which the system effectively and efficiently addresses the information management problem
  - determine whether the system should be continued, enhanced, or replaced/archived
- **Archive:**
  - end the operation of the system in an orderly manner
  - ensure system components and data are properly archived or incorporated into other systems

Decisions

- Production approach decisions include:
  - what evaluations of the system and data should be conducted
  - what new or additional user support activities are needed
- Production execution decisions include:
  - what changes or enhancements are needed
  - should an enhancement be made within this stage or given its own life cycle
- Evaluation approach decisions include:
  - how well does the expert system solve the information management problem

New Products

- Performance Report
- Post Implementation Evaluation Report
- System Evaluation Report
- System Disposition Report

Figure 8.1
Operation Phase
Objectives, Decisions, and Products

The third objective is to develop and implement maintenance changes and minor enhancements. All maintenance and minor changes are controlled through baselines, particularly in the case of expert systems.

8.1.2 Evaluation

The first objective of the Evaluation Stage is to determine the extent to which the system effectively and efficiently addresses the information management problem. Efficient operation, effective management, and current functional and data requirements are all considered in this stage.

The second objective is to determine whether the system should be continued, enhanced or replaced/archived. This decision requires appraisal of advances in technology and estimation of cost/benefit to upgrade the system.

8.1.3 Archive

The first objective of the Archive Stage is to end the operation of the system in a planned orderly manner. Care must be taken not to disrupt OSWER programs that use the system as well as other systems that use the data and/or software of the current system.

The second objective is to ensure that system components and data are properly archived or incorporated into other systems. Resources dedicated to the system being deactivated should be reallocated to functioning systems where appropriate. Data and software that are not currently needed by other systems should be archived against future needs.

8.2 DECISIONS

Many of the decisions made during this phase of expert system development are identical to those made for conventional systems. Only special expert system development considerations are cited below. If additional detail is required, refer to the LCM Guidance.

8.2.1 Production

8.2.1.1 Approach Decisions

The Production Stage deals with two approach decisions. First, what evaluations of the system and data should be conducted? Many factors may affect the selection of specific evaluations including new requirements, system performance, new technology, and cost experience (i.e., costs accrued over the development life cycle). Because expert system technology in particular is subject to rapid evolution, system developers must stay abreast of advances in AI tools and techniques.

Second, what new or additional user support activities are needed? Much will be learned about the required level of system maintenance and user training as the system is developed. Because improper use of an expert system could have serious repercussions, these topics should be monitored carefully to ensure that the system is used properly.

8.2.1.2 Execution Decisions

There are two execution decisions. First, it must be determined what changes and/or enhancements to the system and database are needed.

Second, should a particular enhancement be implemented within this stage, or given its own life cycle? Potential factors to be considered which tend to require the start of a new life cycle include:

- The processing of additional information,
- Major impacts on the system,
- High cost of accomplishing the change, or
- Significant impact on multiple OSWER, regional, or state offices.

Review and approval thresholds will play a major role in determining what action to take. The capacity of the expert system software to be modified will influence this decision.

8.2.2 Evaluation

The Evaluation Stage deals with what changes or improvements in system and data functionality, quality, and/or performance are needed. This approach decision made during this stage will determine how well the system solves the information management problem. Information access, processing, and content are all considered. Expert systems will require special attention in evaluating changes to the knowledge embedded within the system (e.g., as regulations or allowance changes).

8.3 SUCCESS FACTORS

Several factors that can impede success if not considered in the Production, Evaluation, and Archive Stages are described below. This Section focuses on perceptions of what expert systems can and cannot do and offers solutions or alternatives.

8.3.1 Production

In the Production Stage, users should not become overly reliant on an expert system's recommendations. Because the advice offered by the system is so similar to the human expert's recommendations, the advice may be accepted as infallible. This is particularly dangerous when the system is working with incomplete or incorrect information. Users should maintain an attitude of skepticism in proportion to the consequences of the system rendering incorrect advice.

8.3.2 Evaluation

In the Evaluation Stage, the organization should be cognizant of changes that affect the system's performance and should not treat them too casually. Relevant changes such as development of new techniques or expertise, and enhancements in hardware and software may all affect the system. Formal evaluations of all perceived changes and their impact on the system are required.

Translating changes into modifications to the knowledge base will require an iteration of knowledge engineering with the expert. Depending on the extent of the changes, this iteration could almost be a mini-project development effort.

Over time, the problem that the system was developed to address may cease to exist or may be subsumed in a larger problem addressed by another system. "Pride of ownership" can inhibit shutting down a system that is no longer required. This is obviously wasteful and inefficient. All programs and systems should be viewed objectively.

8.3.3 Archive

In the Archive Stage, it is important to retain components of the system that are useful at a later date. In general, knowledge is always valuable. Knowledge in automated form is easily archived and retrieved as needed. However, it is also easy to overcompensate for the first by retaining everything. If an information management problem ceases to exist, significant portions of the solution to the problem are probably discardable. Large systems may consume considerable physical and logical storage space; both are expensive and should be used efficiently.

8.4 PRODUCTS

Figure 8.1 also graphically illustrates the new products associated with the Operation Phase. They are similar to the ones cited in the *LCM Guidance*. Each stage's products are listed below.

8.4.1 Performance Report

The Performance Report describes the experience of system, knowledge base, and database users during Production, noting unanticipated events and potential problems. This report serves as a diagnostic tool to aid the project manager, as well as assisting evaluations of the system, knowledge base and database(s). This report is usually brief, and may include extracts from computer facility reports that identify the resources used by the system and database(s).

8.4.2 Evaluation

8.4.2.1 Post-Implementation Evaluation Report

The Post-Implementation Evaluation Report presents a complete assessment of the implemented system based on the experience of the initial period of system operation. This report addresses all facets of the system, including degree of satisfaction of functional and data requirements, technical performance, and system management. It also identifies potential new requirements not addressed by the system. Specific recommendations are provided, where appropriate, to help ensure that the system continues to respond to the identified information management problem. With regard to expert systems, particular attention should be paid to new or changing functional and data requirements, ongoing training, and database and knowledge base management.

8.4.2.2 System Evaluation Report

The System Evaluation Report presents the results of a formal assessment of the system. The assessment may vary in scope, focusing on how well the system addresses the information management problem, technical performance of the system, and/or system management practices. The report provides specific recommendations where appropriate and notes those recommendations approved by program management for action. If the evaluation is conducted by a completely independent third party, the evaluation should include a Section including the opinion of the project team with regard to the findings and recommendations of the evaluation.

8.4.3 System Disposition Report

The System Disposition Report:

- Describes the rationale for ceasing system operations,
- Documents the plan for ceasing operations and effectively archiving the various components of the system, and

- Provides information about the location of archived materials.

This report is vital to ensure that information about the system can be accessed to support reactivation of the system, or future reuse of portions of the current system by other systems.

8.5 ACTIVITIES

In addition to activities resulting in the creation of the products listed in Section 8.4, the project management and development team should conduct activities to resolve human resource issues, ensure the successful completion of the production stage, maintain the system, and evaluate the system. These activities are described in the following Sections.

8.5.1 Human Resource Activities

8.5.1.1 Soliciting Feedback from the Users

The users of the expert system play the most important role in the Production and Evaluation Stages. By this time the expert system is a fully working, usable system and the user can thoroughly critique its performance. They are best able to determine if the system is efficient and effective. It is important to gather the user's comments and recommendations at this phase to evaluate if the system should be continued, enhanced, or archived.

8.5.1.2 Two-Way Communication with Management

Management also has a heavy involvement in these stages. If the users desire to continue using the system, they must have management's support to allocate funding and resources. Likewise, management must evaluate the user's requests and comments and incorporate them so that the system maintains its effectiveness.

8.5.2 Production Activities

Project managers and software developers must also consider distribution and end-user support for the completed system. The distribution issues are presented in Section 8.5.2.1; the end-user support issues in Section 8.5.2.2.

8.5.2.1 Distribution

8.5.2.1.1 Configuration Management

Formal review of requested changes to the system before they are made is essential to the integrity of the system.

The procedures for reviewing requested changes are contained in the Configuration Management Plan (a part of the overall Project Management Plan).

8.5.2.1.2 Software Disclaimers

Software disclaimers use limited warranties to restrict vendor and manufacturer liability to replacement of faulty software. They typically deny responsibility for the consequences of improper use of the software or for use of outdated versions. Because the advisory capacity of expert systems presents new implications for improper system development or use, any software disclaimers by vendors, manufacturers, or developers should be carefully studied. Standard disclaimers will probably emerge in the near future as expert systems applications become "off-the-shelf" products.

8.5.2.2 End-User Support Requirements

8.5.2.2.1 Ongoing Training

Providing ongoing training is important in the Production Phase. New users will require training to effectively use the system, and experienced users will require retraining whenever significant changes are made to the system. Systems that generate critical output may require regular certification of all users.

8.5.2.2.2 Documentation

All changes should be completely documented to provide system users and those responsible for operating and maintaining the system the information they need to properly use the system and to perform other activities for effective system use.

Effective reactivation of the system in the future will depend heavily on having complete documentation. It is generally advisable to archive all documentation, including the life cycle products generated during the earliest stages of the life cycle, as well as the documentation for users and for operation and maintenance personnel.

8.5.2.2.3 User Groups

Manufacturers of larger expert system shells support user groups in their major client bases. These groups provide

a forum for exchanging experiences with the tool and with particular applications of it. All organizations should be represented at user group meetings.

8.5.2.2.4 User Feedback

User feedback helps to determine what enhancements to the system are needed to continue to solve the information management problem, including requested changes deferred from prior stages, and developing and implementing these enhancements.

8.5.2.2.5 Software Updates and Upgrades

Software updates and upgrades are necessary in the Production Phase to ensure a successful evaluation for the continued use of the system. Any new releases of system software and applications software packages to operate the system should be provided.

8.5.3 Evaluation Activities

Evaluation considerations that must be considered by the project manager or software developer are maintenance for the system and a post implementation evaluation. These two considerations are described in the following Sections.

8.5.3.1 Maintenance

During the Evaluation Stage, potential changes to the system are considered to ensure that the system continues to solve the information management problem. Changes may take the form of routine maintenance or may constitute formal enhancements to the system or databases.

8.5.3.1.1 Maintenance and Revalidation Options

There are three major sources of change proposals and maintenance activities: users, experts, and the MIS department.

- Users typically request minor enhancements that provide additional functionality and/or improve performance but do not alter the knowledge or data structures. They provide maintenance support by putting any changes through "real world" testing. Experienced user criticisms and comments should be considered carefully: they know best how the system works and does not work.

- After an expert system has been developed and fielded, the problem domain may change for technical, regulatory, or procedural reasons. The experts will probably be the first to recognize the impact of these changes on the system. They should therefore be regularly involved in assessing how well the system continues to address the problem. They will also assist in revalidating the system after any change is implemented by evaluating the effects of the change on other parts of the system.

- In most organizations, the ADP department is responsible for actually implementing changes to supported systems. Just as in system development, MIS staff will need to understand expert system concepts in order to properly test and validate after making changes. They are also an excellent source of knowledge about system capacity and load patterns.

8.5.3.1.2 Knowledge Maintenance

If the system has captured all the relevant knowledge in the task domain and if that knowledge will not change in the foreseeable future, then knowledge maintenance will not be needed. Few problem domains, however, are totally static. If an expert system has been developed specifically because the task domain is changing, then users, experts, and the MIS department will all participate in a prolonged Production Phase that will incorporate multiple Evaluation Stages.

8.5.3.2 Post-Implementation Evaluation

The system should be reviewed formally to determine whether it is operating correctly and efficiently from a technical standpoint, and whether it continues to effectively address the information management problem. This formal Post Implementation Evaluation is conducted only once, during the first occurrence of the Evaluation Stage. The project manager or software developer should evaluate user satisfaction and task performance as well as prepare a benefit analysis during the Post-Implementation Evaluation.

8.5.3.2.1 User Satisfaction

The degree of user satisfaction will be reflected in the Post-Implementation Report.

8.5.3.2.2 Task Performance

The system should be evaluated for task performances against the system specifications. The Post-Implementation Evaluation is used to perform a comprehensive appraisal of the system. This evaluation addresses all tasks of the system: support of functional and data requirements, system technical performance, and effectiveness of system management. All specialized expert system test, review, and validation procedures used in developing the system are now used to evaluate it.

8.5.3.2.3 Benefits Analysis

For each evaluation in this stage a decision is made regarding the future of the system. If it is operating correctly and effectively, the Production Stage continues. This decision requires appraisal of advances in technology and estimation of cost/benefit to upgrade the system. If the system is no longer operating correctly or effectively, and improvements would not be cost-effective, the life cycle proceeds to the Archive Stage.

APPENDIX A
EVALUATING RULE-BASED EXPERT SYSTEM SHELLS

A.1 INTRODUCTION

This appendix describes a set of criteria for evaluating and comparing rule-based expert system development environments (or shells). These guidelines will assist expert system developers in making an informed, objective decision when selecting a rule-based shell that is well-suited to their development objectives and requirements.

Commercially available expert system shells are proliferating; every major and minor software developer seemingly is in the market. Prices of the various packages range from less than $100 to almost $100,000 -- the majority are sold for less than $5,000. The capabilities and quality of the shells cover almost as broad a range, but not in direct proportion to price. With the constant emergence of new packages and frequent upgrades to older ones, the market is extremely volatile; buyers must beware of poorly supported or otherwise inferior software. Since there are no standards against which to evaluate a particular shell and different shells are best suited to different types of problems, a strategy must be developed to address the problem.

The evaluation of expert system shells differs from that of languages and other programming tools because of the lack of standards. The challenge for the system developer is to carefully consider the characteristics of his project and then match those characteristics to a shell that is appropriate. If the type of problem to be addressed by the shell is known, then a small benchmark rule base can be designed for implementation on the various shells being evaluated. In this way a standard of comparison can be loosely set. If the problem type is unknown or varied, then the shell must be evaluated in a more abstract fashion.

This appendix describes the major factors that should be considered as part of a comprehensive evaluation of rule-based expert system shells. Two attachments accompany this appendix: the first is an evaluation form that incorporates the major evaluation factors in a convenient checklist; and the second is a comparison of expert system application types to shell characteristics that is included to aid the developer in determining what type of shell is best suited to a given problem. The conclusion provides some general comments and recommendations to guide expert system developers in the evaluation process.

A.2 SHELL EVALUATION FACTORS

This Section briefly describes the important factors that should be considered in the comprehensive evaluation of a rule-based expert system shell. These factors have also been listed in Attachment A.1, Expert System Shell Evaluation Form. In evaluating a particular shell, the evaluator should quantify whatever values possible in order to enhance the objectivity of the evaluation. The descriptions below attempt to summarize the types of issues that should be addressed in each Section of the form.

A.2.1 General Information

Attachment A.1, Sections 1 through 4 provide an outline for recording general information. Since the shells being considered may be reviewed by more than one person or by persons unknown to each other, record should be kept of the evaluators. Vendor contacts should be listed in order to provide a source for further purchase or evaluation information. Any impressions of the vendor should also be noted, such as how long the shell has been on the market or general impressions about the quality of the vendor's product line. Hardware and software requirements should be noted also. Attention should be paid to the minimal and recommended configuration and the actual configuration used to test the shell.

A.2.2 Features

A survey of the features present in the shell is an important initial step in assessing the shell. The key features that should be evaluated are described below. These features are listed in Section 5 of Attachment A.1.

A.2.2.1 Structure (Attachment A.1, Section 5.1)

Different software tools use different types of parameters to represent values. Many, such as logical and string variables, are commonly supported by expert system shells, but some common ones, such as arrays and floating point numbers, frequently are not. In order to determine which data types are required, the evaluator must fully define the problem domain in advance. If that is impossible, then all data types that may possibly be needed should be considered.

Flexibility in the formation of rules varies widely among shells. Some do not permit multiple clauses in the

premise (IF) Section of the rule, or make allowance for an alternate conclusion (ELSE, ELSE IF). Once again, the particulars of the problem to be addressed will dictate what rule structure is needed, but greater flexibility will permit more general application of the package.

Many expert systems must be capable of handling uncertain or incomplete data. The method used by a particular shell to address uncertainty is an issue of concern. How does the shell treat confidence factors (if at all)? Does the shell permit probabilities only in the rule premise? What formula does it use to combine probabilities? What scales (e.g., -1 to +1, 0 to 100) are available?

Some applications may require extensive mathematical capabilities. The accuracy of mathematical calculations and comparisons will vary depending on the internal representation method; a variable may display a value of 0.000 but still not act as a zero flag because its actual value is 0.0001.

Frequently expert systems rely on external software to perform data calls and mathematical operations. A desirable shell feature is the ability to link to popular applications packages such as 1-2-3 or dBASE III, or access routines written in programming languages such as C, Pascal, or assembler. Another consideration is whether the shell can access the underlying operating system and utilities such as the system clock.

The type of problem to be addressed generally determines the type of knowledge representation to be used. There are problems that may be addressed in several ways and the differences between the methods of knowledge representation are sometimes subtle. Rules, frames, and scripts are the most widely used representation methods, but other obscure methods are sometimes encountered. Rule-based shells are the most easily learned and their knowledge bases are the simplest to understand.

Two methods of reaching conclusions are generally found in shells: forward chaining and backward chaining. Forward chaining shells are best suited to data driven problems. Backward chaining shells are best suited to diagnostic problems wherein a hypothesis is tested by stepping backward through the rules. It is important to note that either search method can be implemented in the other, although not very efficiently. Some packages are

capable of both methods of search, either in turn or simultaneously.

Finally, an important feature to consider is the overall size of the shell. The number of rules that it can manipulate in one knowledge base is a quick measure of this, but it is sometimes a false measure. For example, a package that can work with 2000 rules sounds superior to a package that can work with 1000 rules until the user realizes that the first package has no alternate conclusion or multiple premise capabilities. The second system can thus store the same amount of knowledge in a much smaller knowledge base. Another factor to consider is the ability to link smaller knowledge bases together in order to exceed the inherent limit of the shell.

A.2.2.2 Creating the Knowledge Base (Attachment A.1, Section 5.2)

Access to the knowledge base as it is being created is critical. Many shells have an interactive editor built in that provides syntax and variable scope checking as the rules are created. Some shells provide access only through an external text processor. The ability to store input and output in test files can tremendously simplify testing a system, especially when the user input is long or tedious.

Debugging features also vary widely. Error messages can be either insightful or cryptic, depending on the particular shell. Identification of the location of the error (not just reporting that the error exists) and backward and forward tracing are variably supported. A feature seldom found but of great value is the identification of unreachable goal states and unusable rules.

A.2.2.3 Processing the Knowledge Base (Attachment A.1, Section 5.3)

Once the knowledge base has been created and made accessible to a larger audience, it will have to provide explanations to that audience of how and why it reaches particular conclusions. The text displayed upon request and the points at which such requests can be made are important issues to be considered. Users will also probably want a "what if" or "undo" capability in order to explore the consequences of changing decisions.

The performance of an expert system shell can be evaluated in several ways. A set of standard benchmarks should be developed to test rule processing speed, capacity, and relative development time. Typical benchmarks include the Tower of Hanoi problem and the animal classification game. It is difficult to compare the performance of shells in any great detail, but these or similar tests give a rough indication of relative performance.

Performance can also be gauged approximately by determining whether the shell is compiled or interpreted. Each has its advantages and disadvantages; the particular needs of the user will determine which is better for a given problem. The shell's documentation may also give a measure of performance, possibly in the form of logical inferences per second (LIPS).

A.2.2.4 Portability (Attachment A.1, Section 5.4)

Portability of expert systems can be addressed from several perspectives. The value and usefulness of knowledge bases is greatly enhanced if they are represented in an easily comprehensible manner. If they are stored in a generic form (e.g., an ASCII file), then they can be ported across word processing packages and displayed in a comprehensible form. If the rule structure is sufficiently English-like (as opposed to complex computer code), it can readily be interpreted by a non-programmer expert. While the knowledge bases produced by most shells cannot be directly ported to other shells, the ease with which they can be converted to other environments should be considered. In particular, shells written in the same language may be more compatible than shells written in different languages.

A.2.2.5 User Interfaces (Attachment A.1, Section 5.5)

How the development and delivery systems interact with the developer and user are significant issues. The use of pull-down menus, the availability of on-line help, and a transparent operating system are among the many desirable features that facilitate interfacing with the system. Graphic representation of the decision process, either in static or dynamic form, can tremendously enhance the ability of the system to present information to the user.

A.2.2.6 Documentation (Attachment A.1, Section 5.6)

As with any other type of software, good documentation of shells is critical. The quality of the tables of contents, indices, illustrations, and examples in the developer's and user's references will drastically affect the ability of the developer and user to learn the shell and solve the problems that inevitably arise. On-line or hard copy tutorials are useful features that can significantly reduce training time.

A.2.2.7 General, Explanations, and Additional Features. (Attachment A.1, Sections 5.7 - 5.9)

Because of the wide variety of features offered by different shells, no fill-in-the-blanks form can completely describe the attributes of all shells. Therefore, the evaluator should make general comments about the suitability of a given shell. Attachment A.1, Expert System Shell Evaluation Form, provides ample room for explanations of other features or descriptions of other problems. This is sometimes the most important part of an evaluation; the evaluators should be liberal in their note taking.

A.2.3 Overall Evaluation and Comments (Attachment A.1, Sections 6 and 7)

Finally, a conclusion has to be drawn about the overall quality of the shell and its suitability to the problem at hand. This is ultimately a subjective opinion since the evaluators may have certain biases. Therefore, if one evaluator is going to review each of several packages being considered for a particular application, the same individual should evaluate them all in order to permit a reasonably fair comparison. If time permits several evaluators to review each shell, then a higher degree of objectivity can be attained.

A.3 CONCLUSION

The evaluation of a rule-based shell can be divided into a review of the overall structure; the creation, processing, and portability of the knowledge base; user interfaces; documentation; and other miscellaneous information. This information should be recorded on a standard form, such as provided in Attachment A.1, in order to facilitate comparisons of different rule-based shells. However, the evaluator must be careful not to get caught up in the necessity of filling in every

single blank on the form and thus lose sight of the larger objective of getting a feel for the suitability of the shell to a given problem. Ultimately the question to be answered is, "Does the package do the job as intended?" Such fundamental questions as "Does the package work as explained in the manual?" and "Does the package have unexplained features?" are frequently overlooked because they are difficult to quantify.

Many commercial expert system shells were developed to solve one specific type of problem and have since been modified into general problem solvers. The features added to make the shell a general purpose tool are sometimes very obviously add-ons; the underlying functionality is not enhanced by them. The evaluator must seek out the underlying functionality and not be misled by the peripheral features. Remaining objective in this manner is essential to a proper evaluation. Attachment A.2, Shell Characteristics vs. Application Types, correlates the shell features discussed here to typical expert system application types as discussed in Chapter 1.

Attachment A.1
EVALUATION FORM
FOR RULE-BASED EXPERT SYSTEM SHELLS

1. TESTER

Name: _____ Date: _____

Office: _____ Phone: _____

2. PRODUCT

Name: _____ Price: _____

Release/Version: _____ GSA (Y/N)? _____

Number of Installations: _____

Run Time Version Price: _____ User Groups: _____

3. VENDOR

Name: _____

Address: _____

_____

_____

Years in Business: _____ Financial Status: _____

Support Available: _____ Training Available: _____

4. REQUIREMENTS

Minimal	Recommended
Hardware: _____	Hardware: _____
_____	_____
_____	_____

Operating System: _____

Systems it will run on: _____

_____

System tested on: _____

5. FEATURES

If a feature is present, place an X in the box next to it. If an explanation is necessary, place a number in the box and add an explanation in the EXPLANATIONS section. If the feature is not present and no explanation is needed, leave the box unmarked. An asterisk indicates a core feature.

5.1 STRUCTURE

Types of parameter values

* Logical ☐

* Numeric ☐

* Enumerated ☐

Floating point ☐

Complex rule structures

Premise (IF) section

Multiple ANDs ☐

Multiple ORs ☐

Conclusion (THEN) section

Multiple ANDs ☐

Confidence factor (deal with uncertainty)

Bayesian probability ☐

MYCIN formula (CF1+CF2/100*(100-CF1)) ☐

Other (specify) ☐

Multiple rule contexts ☐

Rules refer to parameters in direct contexts ☐

Math capability to combine or evaluate rule contexts ☐

Linkage to higher level programming languages (HLPL)

Use HLPL to create values or get data ☐

Use HLPL to combine parameter values ☐

Use HLPL to perform mathematical operations ☐

Variety of knowledge representation forms

* Rules ☐
Frames ☐
Examples ☐
Other (specify) ☐

Chaining mechanism

Forward only ☐
Backward only ☐
Both ☐

* Capacity (number of rules)

Object-oriented structure ☐
Extensible expert system shell (explain) ☐

5.2 CREATING THE KNOWLEDGE BASE

Interactive rule-building with error checking ☐

* Test file processing

Save an interactive session and replay it ☐
Run a test file ☐

* Debugging features

Meaningful error messages ☐
Help in locating the source of the problem ☐
Interaction between run, debug, and edit ☐
Check for unused rules ☐
Trace rule sequencing/conclusion building ☐

5.3 PROCESSING THE KNOWLEDGE BASE

* Explanation capability

Why a question is being asked ☐
How a conclusion is reached ☐
Other (explain) ☐

Undo capability ☐
Truth Maintenance ☐
Performance
Compiled ☐
Interpreted ☐
Good response time (explain) ☐

5.4 PORTABILITY

Knowledge base stored in ASCII file (or other standard) ☐

5.5 USER INTERFACE

* Invisible operating system ☐
Menus ☐
Commands ☐
On-line help ☐
* Logical, intuitive operation ☐
* Protects against user errors ☐
* Provides meaningful error message ☐
Graphics display capability
Show decision structure ☐
Show data structure ☐
Illustrate parameter or derived values ☐
Show relations between parameters ☐
Other ☐
Mouse, icons, pop-up menus, windows ☐

5.6 DOCUMENTATION

Easy-to-use table of contents ☐
* Well organized ☐
* Well written ☐

Good examples..... ☐
Illustrated..... ☐
* Good index..... ☐
Programmer's reference..... ☐
* User's reference..... ☐
Tutorial..... ☐

5.7 GENERAL

Bug-free (verified by extensive use)..... ☐

5.8 EXPLANATIONS

5.9 ADDITIONAL FEATURES

6. OVERALL EVALUATION

Does this package meet your expectations for a product of its type?	☐ Yes	☐ No
Does this package meet all the core requirements in the evaluation form?	☐ Yes	☐ No
Do you feel that this package should be listed as a preferred package	☐ Yes	☐ No

How many hours did it take you to become reasonably proficient in the package's use? _____ hours

The following items are rated on a scale of 1 to 10. A poor rating is indicated by 1 and an excellent rating is indicated by 10.

Performance _____	Documentation _____	Recommendation _____
Ease of Use _____	Ease of Learning _____	Flexibility _____

7. COMMENTS

Attach...nt A.2
Shell Characteristics vs. Application Types

● Probably Required
○ Possibly Required
❑ Not Likely to be Required

	Diagnosis & Classification	Data Analysis & Interpret.	Design & Synthesis	Prediction & Simulation	Monitoring	Use Advisor	Intelligent Assistant	Planning & Scheduling	Control
Inference Approach									
Backward Chaining	●	○	○	❑	○	○	○	○	○
Forward Chaining & Reasoning	○	●	●	●	●	●	○	●	●
BC, FC, & Forward Reasoning	●	●	●	○	○	○	●	●	○
Hypothetical Reasoning	●	●	●	○	○	○	○	●	❑
Object Oriented	○	○	○	●	○	○	○	○	●
Blackboard	●	●	●	○	○	○	○	●	❑
Induction	●	○	○	❑	❑	○	○	❑	○
Object Description									
Frames	●	○	●	○	○	○	○	●	○
Frames with Inheritance	●	○	●	○	○	○	○	●	○
Objects	○	○	○	●	○	○	○	○	●
Parameter Values Pairs	○	○	○	❑	○	○	❑	❑	○
Logic	○	○	○	○	○	○	○	○	○
Rules	○	○	○	○	○	○	○	○	○
Certainties	●	●	○	○	○	○	○	○	○
Actions									
Rules	●	●	●	●	●	●	●	●	●
Examples	●	○	○	○	❑	○	○	❑	○
Logic	●	●	●	●	○	○	○	○	○
Messages	○	○	●	●	○	○	○	●	●
Procedures	○	○	●	●	●	○	○	●	●

APPENDIX B

GLOSSARY

AI. See artificial intelligence.

Algorithm. A formal procedure guaranteed to produce correct or optimal solutions.

Architecture. (1) The organizing framework imposed upon knowledge applications and problem-solving activities. (2) The knowledge-engineering principles that govern selection of appropriate frameworks for specific expert systems.

Artificial intelligence. The subfield of computer science concerned with developing intelligent computer programs. This includes programs that can solve problems, learn from experience, understand language, interpret visual scenes, and, in general, behave in a way that would be considered intelligent if observed in a human.

Backtracking. A search procedure that makes guesses at various points during problem-solving and returning to a previous point to make another choice when a guess leads to an unacceptable result.

Backward chaining. An inference procedure or strategy used by the inference engine where the system starts with what it wants to prove (e.g., the goal), and tries to establish the facts it needs to reach that goal. The text of the goal is matched against the conclusion (i.e., in the example "if the sky is cloudy, then the forecast might include rain" is a conclusion) for each rule to determine whether that rule will contribute information to the resolution of the goal. If the conclusion of a rule matches the goal, then the premises of the rule are considered in turn. Each of the premises is considered to be an intermediate goal. Results of evaluating each goal are stored in memory to be used when evaluating subsequent goals.

Baseline. The set of life cycle products and other related documentation which officially comprises the system at a given point in time.

Blackboard. A database globally accessible to independent knowledge sources and used by them to communicate with one another. The information they provide each other consists primarily of intermediate results of problem solving.

Chaining, backward. See backward chaining.

Chaining, forward. See forward chaining.

Conflict resolution. The technique of resolving the problem of multiple matches in a rule-based system. When more than one rule's antecedent matches the database, a conflict arises since (1) every matched rule could appropriately be executed next, and (2) only one rule can actually be executed next. A common conflict resolution method is priority ordering, where each rule has an assigned priority and the highest priority rule that currently matches the database is executed next.

Control structure. Any procedure, explicit or implicit, that determines the overall order of problem-solving activities; the temporal organization of subprocesses.

Domain expert. A person who, through years of training and experience, has become extremely proficient at problem solving in a particular domain.

Domain knowledge. Knowledge about the problem domain, e.g., knowledge about geology in an expert system for finding mineral deposits.

End user. The person who uses the finished expert system; the person for whom the system was developed.

ES. See expert system.

Expert system. A computer program that uses expert knowledge to attain high levels of performance in a narrow problem area. These programs typically represent knowledge symbolically, examine and explain their reasoning processes, and address problem areas that require years of special training and education for humans to master.

Expertise. The set of capabilities that underlie the high performance of human experts, including extensive domain knowledge, heuristic rules that simplify and improve approaches to problem-solving, metaknowledge and metacognition, and complied forms of behavior that afford great economy in skilled performance.

Expert system development environment. The programming language and support package used to build the expert system.

Explanation. Motivating, justifying, or rationalizing an action by presenting antecedent considerations such as goals, laws, or

heuristic rules that affected or determined the desirability of the action.

Explanation facility. That part of an expert system that explains how solutions were reached and justifies the steps used to reach them.

Fact. A proposition or datum whose validity is accepted.

Forward chaining. One of several control strategies that regulate the order in which inferences are drawn. In a rule-based system, forward chaining begins by asserting all of the rules whose IF clauses are true. It then checks to determine what additional rule might be true, given the facts it has already established. This process is repeated until the program reaches a goal or runs out of new possibilities.

Frame. A knowledge representation method that associates features with nodes representing concepts or objects. The features are described in terms of attributes (called slots) and their values. The nodes form a network connected by relations and organized into a hierarchy. Each node's slots can be filled with values to help describe the concept that the node represents. The process of adding or removing values from the slots can activate procedures (self-contained pieces of code) attached to the slots. These procedures may then modify values in other slots, continuing the process until the desired goal is achieved. Also called Data Directed Reasoning.

Heuristic. A rule of thumb or simplification that limits the search for solutions in domains that are difficult and poorly understood.

Inference engine. That part of a knowledge-based system or expert system that contains the specific procedures and algorithms for using the rules/heuristic in the knowledge base to solve a problem. The inference engine processes the domain knowledge (located in the knowledge base) to reach new conclusions.

Knowledge. The facts, beliefs, and heuristic rules a computer program must have to behave intelligently.

Knowledge acquisition. The process of extracting, structuring, and organizing knowledge from some source, usually human experts, so it can be used in a program.

Knowledge base. The portion of a knowledge-based system or expert system that contains the domain knowledge.

Knowledge-based system. A program in which the domain knowledge is explicit and separate from the program's other knowledge.

Knowledge engineer. The person who designs and builds the expert system. This person is usually a computer scientist experienced in applied artificial intelligence methods.

Knowledge engineering. The discipline that addresses the task of building expert systems; the tools and methods that support the development of an expert system.

Knowledge management. The process of formally controlling any changes or additions to the knowledge base in order to maintain expert system integrity.

Knowledge representation. The process of structuring knowledge about a problem in a way that makes the problem easier to solve.

Knowledge source. Generally, a body of domain knowledge relevant to a specific problem. In particular, a codification made applicable for an expert system.

Natural language. The standard method of exchanging information between people, such as English (contrasted with artificial languages, such as programming languages).

PROLOG. An Artificial Intelligence programming language based on predicate calculus. PROLOG is short for the French words Programmation en Logique.

Real-world problem. A complex, practical problem which has a solution that is useful in some cost-effective way.

Representation. The process of formulating or viewing a problem so it will be easy to solve.

Robustness. That quality of a problem solver that permits a gradual degradation in performance when it is pushed to the limits of its scope of expertise or is given error laden, inconsistent, or incomplete data or rules.

Rule. A formal way of specifying a recommendation, directive, or strategy, expressed as IF premise THEN conclusion or IF condition THEN action.

Rule-based methods. Programming methods using IF-THEN rules to perform forward or backward chaining.

Rule-based program. A computer program that constitutes a module of heuristic knowledge.

Search. The process of looking through the set of possible solutions to a problem in order to find an acceptable solution.

Shell. The common term for expert system development environment.

Symbol. A string of characters that stands for some real-world concept.

Symbolic reasoning. Problems solving based on the application of strategies and heuristic to manipulate symbols standing for problem concepts.

Tools for knowledge engineering. Programming systems that simplify the work of building expert systems. They include languages, programs, and facilities that assist the knowledge engineer.

Tree structure. A way of organizing information as a connected graph where each node can branch into other nodes deeper in the structure.

User. A person who uses an expert system, such as an end-user, a domain expert, a knowledge engineer, a tool builder, or a clerical staff member.